UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA
DIVISIÓN DE ESTUDIOS DE POSGRADO

RECONOCIMIENTO DEL PARLANTE UTILIZANDO CUANTIZACION VECTORIAL Y REDES NEURONALES DE TIPO LVQ

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN INGENIERIA
(ELECTRICA)

P R E S E N T A

MARCIAL CONTRERAS BARRERA

DIRECTOR: DR. JESUS SAVAGE CARMONA

CIUDAD UNIVERSITARIA NOVIEMBRE 2002

Dedicatorias y Agradecimientos

A mi Papá y Mamá por enseñarme que la vida se conquista con esfuerzo.

A María por su apoyo incondicional para la realización de este trabajo y la realización de este sueño.

A Jessica Lizbeth porque día a día es el sueño más importante de mi vida.

Al Dr. Jesús Savage Carmona porque ha sido un ejemplo y estimulo muy importante en mi formación académica.

Al Lic. Gonzalo Reséndiz Cansino por sus enseñanzas y el apoyo incondicional para la realización de este trabajo, y sobre todo por confiar en mi.

A Patricia y Mauricio por que con su risa y entusiasmo caminamos juntos en la conquista de la vida.

A mis amigos por compartir los momentos felices de la vida.

Índice

Capítulo 4. Redes Neuronales Artificiales

Capítulo 5. Desarrollo y Resultados

Capítulo 1

Introducción

1.1 Antecedentes

El procesamiento de voz ha sido un área de investigación desde hace varias décadas con una amplia gama de aplicaciones, desde comunicaciones hasta la conversión de texto a voz. Conforme al campo del procesamiento digital de señales se ha desarrollado basándose en la evolución de los sistemas digitales y el desarrollo de los algoritmos de procesamiento digital de señales. Dicho procesamiento se puede clasificar en varias áreas de aplicación que son: reconocimiento de voz, verificación del parlante e identificación, síntesis de voz y texto a voz; por lo tanto los algoritmos de procesamiento digital de voz pueden ser utilizados en cualquiera de estas áreas.

El reconocimiento del parlante es el estudio del desarrollo de sistemas de cómputo que puede identificar o validar a un parlante por su voz. Los beneficios de interfaces de usuarios manejados por voz han sido investigados desde hace años. En la actualidad las aplicaciones en el campo del procesamiento digital de voz continúan creciendo. La verificación automática del hablante está relacionada con la verificación de la identidad de una persona basándose en el análisis de las características de su voz.

El proceso de reconocimiento automático de voz ha alcanzado un gran interés en nuestros días, siendo su principal área de aplicación el campo de la seguridad en la verificación de la identidad de una persona, la cual es utilizada para el acceso a ciertos servicios o lugares. El reconocimiento del hablante, puede ser dependiente o independiente del texto.

El reconocimiento del hablante se lleva a cabo en tres fases:

Fase de registro. Durante esta fase el usuario es registrado en el sistema, dando un número de identificación así mismo se le pide que pronuncie una oración o un conjunto de oraciones las cuales sirven como contraseña. La colección de datos de la voz de la persona es guardada, procesada y almacenada en una base de datos. También se realizan varias repeticiones de la oración para registrar variaciones en cada pronunciación, para realizar ésta operación se requiere del trabajo de una o varias sesiones con el usuario.

Fase de entrenamiento. Este es el procedimiento interno durante el cual el sistema construye un patrón con las características más importantes de la voz del usuario como la forma de hablar y sus variaciones, por ejemplo: la calidad del habla. Este patrón de voz es utilizado como referencia en la fase de verificación. El entrenamiento es realizado por el sistema y no requiere de la intervención del usuario.

Fase de verificación. Después de que el usuario se ha registrado con un identificador y su patrón de voz ha sido creado, el sistema está listo para operar. Este es el modo más importante de operación y se pide al usuario ingresar su ID o contraseña. El sistema entonces realiza la decisión de aceptación o de rechazo. Es importante resaltar que la voz puede cambiar con el tiempo motivo por el cual se deben considerar nuevas muestras de voz para actualizar su patrón y almacenarse en la base de datos.

El sistema de reconocimiento puede ser caracterizado como texto dependiente o texto independiente. El sistema texto dependiente requiere que la prueba del habla contenga el mismo texto que en el proceso de entrenamiento es decir, un parlante entrenará al sistema con una frase llamada "clave de entrada". Entonces en futuras sesiones de verificación el parlante requerirá repetir la misma frase como clave. El sistema texto independiente desarrollará un modelo de parlante basado en un modelo general. La identificación o verificación se puede realizar usando alguna expresión, la cual puede o no ser la misma que la frase clave.

1.2 Definición del Problema

Dadas las diferentes áreas de aplicación y la necesidad de establecer mecanismos que permitan establecer comunicación hombre-máquina, y control de acceso a lugares o información restringidos es necesario el desarrollo de sistemas automáticos capaces de reconocer un conjunto de hablantes independientes del texto almacenado en una base de datos, que sean utilizados para el control de la información de tipo clasificada o bien para el acceso a lugares de tipo restringido.

1.3 Objetivos

Los objetivos de esta tesis son:

- Aplicar los algoritmos de reconocimiento de voz para el desarrollo de un sistema automático capaz de reconocer a un conjunto de hablantes independientes del texto, el cual puede ser utilizado como un sistema de control de acceso a lugares restringidos o a un sistema de información clasificada.

- Evaluar algoritmos clásicos de reconocimiento de voz para el reconocimiento del parlante.

- Desarrollar y aplicar un algoritmo de reconocimiento del parlante basado en algún tipo de Red Neuronal Artificial (ANN) para buscar una mayor robustez en el reconocimiento del parlante.

1.4 Recursos

Los recursos de software y hardware para la realización del proyecto fueron:

- Una tarjeta de sonido Sound Blaster AWE de 64 bits.
- Dos equipos de computo Pentium II.
- Compilador Visual C++ y Matlab.
- La base de datos UNAM-FI con 30 parlantes.

1.5 Alcances

Para lograr el reconocimiento del parlante independiente del texto, se estudiaron diferentes técnicas de reconocimiento de voz como la Cuantización Vectorial, el Análisis Cepstral y las Redes Neuronales. Las técnicas mencionadas anteriormente son las más utilizadas para el desarrollo de sistemas de reconocimiento de voz.

La realización de este proyecto tuvo como finalidad, el desarrollo de programas utilizados en aplicaciones de reconocimiento del parlante, dicho reconocimiento puede ser utilizado en áreas de seguridad para el acceso de información o restricción a ciertos lugares además puede ser implementado en robots que entiendan ordenes e identifiquen a los parlantes con los que se puedan comunicar.

Se desarrolló una técnica de reconocimiento del parlante independiente del texto, basada en un conjunto de vectores de Pitch (tono) y del desarrollo de una red neuronal de tipo LVQ (Learnig Vector Quantization).

Cabe mencionar que los algoritmos de cuantización vectorial empleados, también pueden ser utilizados en el reconocimiento de palabras aisladas dependientes del parlante e independientes del parlante con pequeñas modificaciones.

Capítulo 2

Procesamiento de la Señal de Voz

Introducción

En este capítulo se define el procesamiento digital de la señal de voz, definiendo las características más importantes de la generación de voz, la forma en como son producidos los sonidos, así como la clasificación de los tipos de sonidos (sonoros y no sonoros) y como podemos identificarlos a partir de sus características.

Además se describen algunas herramientas para el procesamiento de la señal de voz, como el preénfasis de la señal para resaltar las altas frecuencias; la aplicación de la ventana de Hamming y la descripción de la función de autocorrelación (herramienta muy utilizada en el procesamiento digital de señales). También se describe la trasformada de Fourier como herramienta de análisis de señales en el dominio de la frecuencia. La aplicación de los algoritmos de procesamiento de la señal de voz da como resultado que el reconocimiento de palabras y del parlante sea una realidad y tenga áreas de aplicación en la actualidad.

2.1 Modelo del Tracto Vocal

El tracto vocal es un tubo terminado en un extremo por la laringe y en el otro por los labios, a través del tracto pulmonar se inyecta aire al tracto vocal pasando por la laringe. En este sistema el sonido se genera de tres formas diferentes. Los sonidos sonoros se producen excitando el tracto vocal con pulsos cuasiperiódicos de aire [2].

Los sonidos fricativos se producen al formar una constricción de algún lugar del tracto, creando así turbulencia que produce una fuente de sonido de banda ancha que excita al tracto vocal. Los sonidos fricativos pueden producirse con o sin fonación [2].

Todas estas fuentes crean una excitación de banda relativamente ancha, el tracto vocal se comporta como un filtro lineal y variante en el tiempo e impone sus propiedades de transmisión sobre el espectro de la fuente. Se ha observado que la variación del filtro es lenta y por lo tanto se considera invariable en un periodo de alrededor de 10 ms. Razón por la cual la fuente de sonido y la forma del tracto son relativamente independientes, por lo que es razonable modelarlas en forma separada, como se muestra en la Figura 2.1

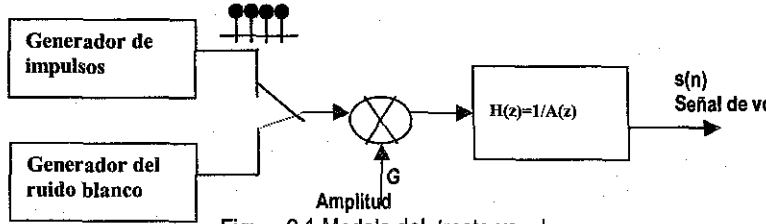


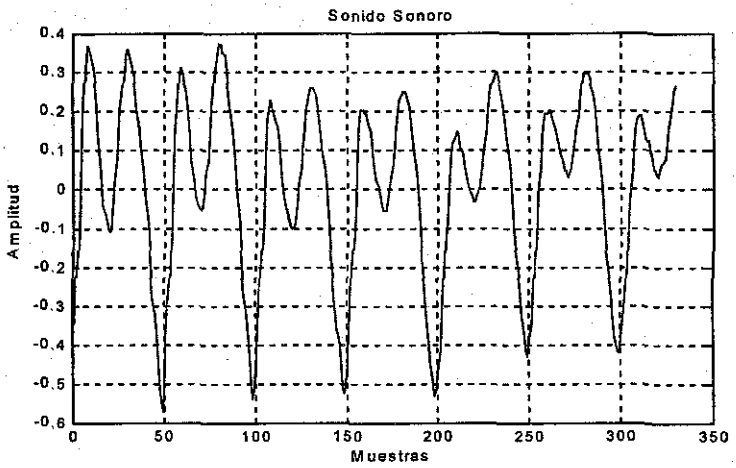
Figura 2.1 Modelo del tracto vocal

Este es el modelo digital en donde las muestras de la señal voz son la salida del filtro digital, variante en el tiempo, que aproxima las propiedades del tracto vocal y cuya excitación es variable. Se trata de un modelo fuente-filtro, llamado así porque modela al sistema como una fuente variable que excita a un filtro variable

En este modelo todas estas propiedades se concentran en un filtro que facilita el modelo del tracto vocal, aunque en realidad representa las propiedades de radiación, del tracto vocal y de los pulsos de fonación.

2.2 Sonidos Sonoros

La característica principal de la señal de los sonidos sonoros es que es cuasiperiódica como se muestra en la Gráfica 2.1

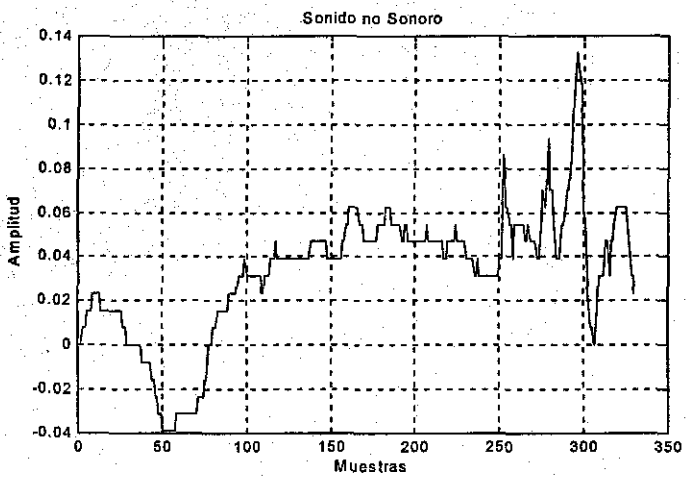


Gráfica 2.1 Sonido sonoro

La vibración de las cuerdas vocales con el paso del aire genera una excitación cuasiperiódica del tracto vocal que modifica el espectro de dicha excitación según su respuesta en frecuencia. Diferentes sonidos cuasiperiódicos pueden tener la misma frecuencia fundamental y sin embargo ser diferentes. Otra característica de estos sonidos es que en general contienen mayor energía que los sonidos sordos. Las formantes determinan el timbre del sonido y tanto su posición como su intensidad dependen de la respuesta en frecuencia del tracto vocal puesto que corresponde a sus máximos [2].

2.3 Sonidos No Sonoros

Los sonidos no sonoros son aquellos que no involucran fonación. Su energía es menor a la de los sonidos sonoros y en general está concentrada en frecuencias altas. Se modelan como procesos aleatorios ya que provienen de procesos turbulentos y corresponden a una excitación ruidosa del modelo fuente-filtro, la Gráfica 2.2 muestra un sonido de este tipo.



Gráfica 2.2 Sonido no sonoro

2.4 Energía de la Señal de Voz

Una de las representaciones más simples de una señal es su energía. En el caso de una señal discreta real x(n), la energía se define como:

$$E(n) = \sum_{n=-\infty}^{\infty} x^2(n) \qquad 2.1$$

Para señales no estacionarias es más apropiado considerar un cálculo de energía variante con el tiempo como:

$$E(n) = \sum_{n=0}^{N-1} \langle w(m)x(n-m)\rangle^2 \qquad 2.2$$

Donde *w(m)* es una ventana que selecciona un segmento de *x(n)* y *N* es el número de muestra de la ventana. Si se desea dar el mismo peso a todo el intervalo analizado, se emplea una ventana rectangular, mientras que si se desea dar mayor peso a la sección central del intervalo se puede emplear una ventana de pesos variables, como una ventana de Hamming. La selección de la longitud de la ventana es muy importante ya que si se selecciona una ventana menor al periodo (en el caso de voz sonora), *E(n)* fluctúa muy rápido dependiendo de los detalles precisos de la forma de onda, en cambio si se selecciona una ventana mayor a varios periodos *E(n)* tendrá muy poca variación y no reflejará las propiedades variantes de la señal de voz. Una longitud de 10-20 ms ofrece buenos resultados para la señal de voz [2].

Una dificultad que existe con la medición de energía descrita en la ecuación (2.1) resulta ser que es extremadamente sensible a niveles altos de señal porque está basada en una cuantización.

Otra característica importante de una señal de voz es su tasa de cruces por cero. En una señal digital, un cruce por cero ocurre entre dos instantes de muestreo *n* y *n-1*.

$$\text{Signo}[x(n)] \neq \text{Signo}[x(n-1)] \qquad 2.3$$

Esta característica permite por ejemplo estimar las posiciones de sonidos sonoros y sonidos sordos. En efecto, la energía de los sonidos sonoros está concentrada debajo de 3 KHz mientras que la energía de los sonidos sordos esta arriba de 3 KHz y por lo tanto la tasa de cruce por cero será más alta para sonidos sordos que para sonidos sonoros. Además, los sonidos sonoros generalmente tienen mayor energía que los sordos, lo que ayuda también en la localización de estos tipos de sonidos.

2.5 Frecuencia Fundamental

La frecuencia fundamental se define como la frecuencia a la cual vibran las cuerdas vocales durante un sonido sonoro. La frecuencia fundamental F_0 es un parámetro, del cual es difícil obtener una estimación confiable a partir de la señal de voz. Normalmente la frecuencia fundamental se encuentra dentro de 50 Hz a 500 Hz para la voz sonora. Para voz no sonora F_0 no esta definida. La frecuencia fundamental es comúnmente llamada frecuencia tono (Pitch en inglés), pero si se desea ser más correcto, el término tono se define como una calidad subjetiva de la voz, la cual está relacionada con la frecuencia fundamental [3].

Existen diferentes métodos para calcular el periodo de tono de la señal de voz y pueden dividirse en métodos en el dominio del tiempo como lo es el método de Autocorrelación, y métodos en el dominio de la frecuencia como el Cepstral, en el dominio de la frecuencia la información espectral y la estructura de las armónicas de la señal de voz son utilizadas para estimar el periodo de tono.

2.6 Procesamiento de la Señal Voz

Se hace necesario para el análisis realizar un procesamiento de la señal de voz. Esto se realiza a través de técnicas que permitan extraer la información acústica directamente a partir de la señal vocal emitida, mediante preénfasis y la aplicación de una ventana de Hamming. Una vez que la señal de voz ha sido procesada, puede ser utilizada para realizar la estimación paramétrica o hacer la extracción del tono para que se realice el reconocimiento de palabras o el reconocimiento del parlante [2].

2.6.1 Preénfasis

Un aspecto importante para la estimación de los parámetros de predicción, es la necesidad de tomar en cuenta las componentes de alta frecuencia de la voz ya que en magnitud son menos significativas que las componentes de baja frecuencia; lo cual daría como resultado que el método de resultados satisfactorios en baja frecuencia, mientras que en altas frecuencias tienen un desempeño muy pobre. Para evitar este efecto, se filtra la señal a través de un filtro de énfasis, el cual enfatiza las altas frecuencias antes de la estimación de los parámetros. El filtro de énfasis está dado por la ecuación:

$S^{\circ}(n)=s(n)-0.937s(n-1)$ 2.4

Donde S^{0} es la señal con preénfasis y *s(n) es* la señal de entrada.

Si se está realizando síntesis de voz a la salida del sintetizador, la señal debe ser deénfatizada mediante la siguiente ecuación:

$s(n)= S^{\circ}(n)+ 0.937s(n-1)$ 2.5

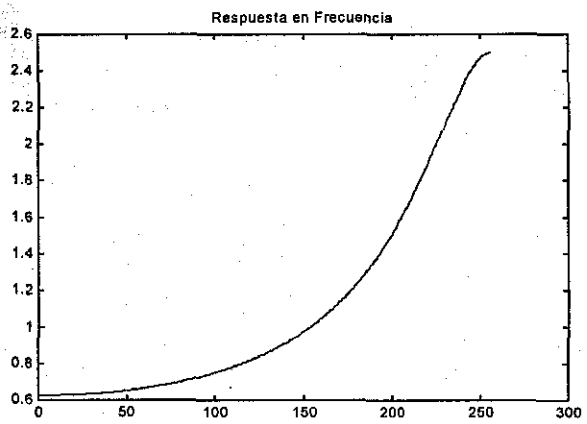
Esta etapa se realiza con el propósito de suavizar el espectro y reducir las inestabilidades de cálculo asociadas con las operaciones aritméticas de precisión finita. Se usa un filtro digital de primer orden cuya función de transferencia es:

$H(z) = 1 - aZ^{-1} \quad ; \quad a = 0.9$ 2.6

Cuya ecuación en diferencias es

$S(n) = s(n) - a*s(n-1)$ 2.7

La Gráfica 2.3 muestra la respuesta en frecuencia del filtro de preénfasis en donde observamos que se comporta como un filtro paso altas.

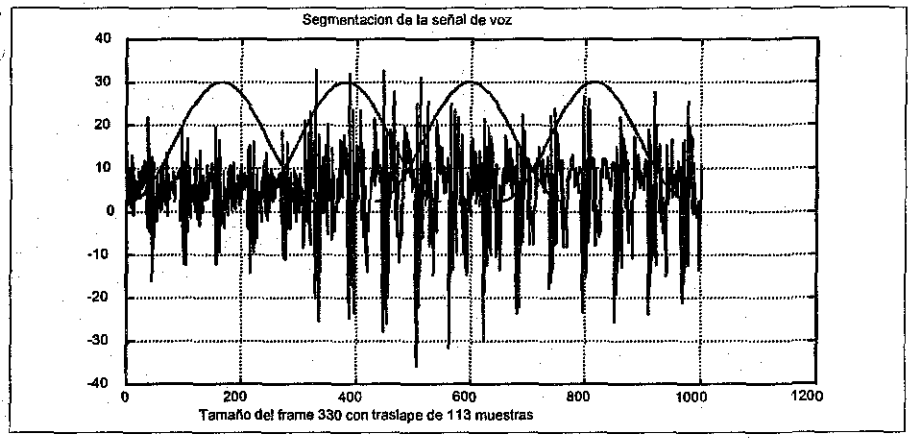


Gráfica 2.3 Respuesta en frecuencia del filtro de preénfasis

2.6.2 Ventana de Hamming

El método de Código de Predicción Lineal (LPC por sus siglas en inglés) funciona bien en señales cuyo comportamiento no cambia con el tiempo, sin embargo éste no es el caso de la señal de voz, para poder utilizar el método *LPC* en el procesamiento de la señal de voz, la señal se segmenta en pequeños segmentos llamados marcos, los cuales son cuasi-

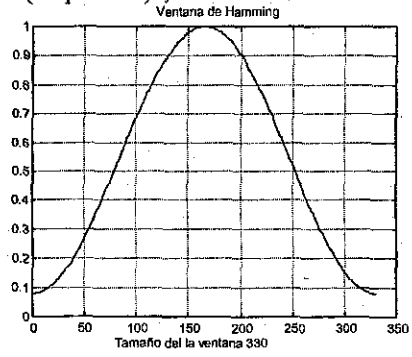
estacionarios, la Gráfica 2.4 muestra un ejemplo de segmentación de la voz. La segmentación se realiza multiplicando la señal de voz *s(n)* por una ventana *w(n)*.



Gráfica 2.4 Aplicación de la ventana de Hamming a la señal voz

La señal con preénfasis se toma cada 10 ms. por espacio de 20 ms. y se le aplica a una ventana de Hamming con el objeto de suavizar la señal en los bordes de dicha ventana. Esta es la ventana que generalmente se usa para el análisis de señales de voz, la Gráfica 2.5 muestra está función y define como:

$$W[nT] = 0.54 - 0.46\cos(2 * pi * n/N), \quad 0 \le n \le N-1 \qquad 2.8$$

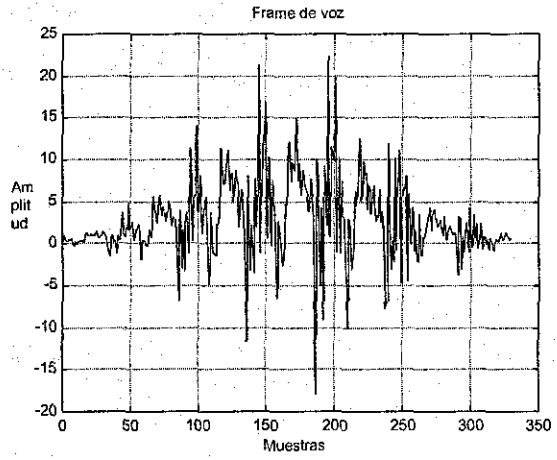


Gráfica 2.5 Ventana de Hamming

Donde *N* es la longitud de la ventana en muestras y es generalmente seleccionada dentro del rango de 20-40 ms, con 30 ms siendo el valor típico [5].

Por ejemplo si consideramos muestras de voz a 8 KHz(8000 muestras/s) el valor típico de la ventana es *N=240* muestras, las ventanas sucesivas son traslapadas y la distancia entre ventanas es llamada periodo del marco, comúnmente el periodo del marco es de 10 a 30 ms (un periodo del marco pequeño garantiza una mejor calidad porque se capturan mejor las

variaciones de la señal voz). La Gráfica 2.6 muestra un marco de la señal de voz después de aplicarle la ventana de Hamming.



Gráfica 2.6 Segmento de voz con la Ventana Hamming

2.7 Autocorrelación

La autocorrelación es un indicador del grado de relación que existe entre dos variables aleatorias. La función de autocorrelación de una señal es una manera conveniente para visualizar ciertas propiedades de la señal [2]. Por ejemplo, si la señal es periódica con periodo *P*, entonces es fácil mostrar que:

R(k)=R(k+P)

Si se tienen dos señales *x(n)* e *y(n)*, de energía finita. La correlación de *x(n)* e *y(n)* se define como:

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-1) \qquad l = 0,\pm1,\pm2,\cdots \tag{2.9}$$

De manera equivalente

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n+l)y(n) \qquad l = 0,\pm1,\pm2,\cdots \tag{2.10}$$

El índice *l* representa el atraso, el subíndice *xy* en la secuencia de correlación $r_{xy}(l)$, representa las secuencias que son correlacionadas. El orden de los subíndices, *x* precedido a *y*, indica la dirección en la cual una secuencia es desplazada con respecto a la otra.

Si invertimos el orden de los índices *xy*, obtenemos:

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n)x(n-1) \qquad l = 0,\pm1,\pm2,\cdots$$

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n+l)x(n) \qquad l = 0,\pm1,\pm2,\cdots \qquad 2.11$$

Al comparar las expresiones anteriores se concluye que

$$r_{xy}(l)=r_{yx}(-l) \qquad 2.12$$

Por lo tanto $r_{yx}(l)$ provee exactamente la misma información que $r_{xy}(l)$ con respecto a la similitud existente entre las secuencias $x(n)$ y $y(n)$. Un caso particular ocurre cuando $y(n)=x(n)$, a la cual se define como secuencia de autocorrelación y queda definida como:

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l) \qquad l = 0,\pm1,\pm2,\cdots \qquad 2.13$$

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n+l)x(n) \qquad l = 0,\pm1,\pm2,\cdots \qquad 2.14$$

Por lo tanto la función de autocorrelación de una señal periódica también es periódica con el mismo periodo [2]. Las propiedades importantes de la autocorrelación son:

- Es una función par Rx(l)=Rx(-l).
- Tiene su valor máximo en l=0
- La cantidad R(0) es igual a la energía para señales determinísticas o potencia promedio para señales aleatorias o periódicas.

Así que la función de autocorrelación contiene la energía en casos especiales, pero la característica más importante es que representa la periodicidad de una señal, para señales periódicas la función de autocorrelación tiene sus máximos en las muestras 0,+p,+2p,..., por lo tanto con el máximo de la función de autocorrelación puede utilizarse para localizar el periodo. Esta propiedad hace a la función de autocorrelación atractiva para localizar la periodicidad en señales incluyendo la voz.

Para realizar la autocorrelación de una señal de voz es más conveniente utilizar la ventana rectangular en lugar de la ventana de Hamming ya que la ventana rectangular resalta más la periodicidad de la señal. Además el tamaño de la ventana que se utilice debería ser por lo menos 2 periodos de la señal que se esté analizando.

2.8 Transformada de Fourier

El análisis en frecuencia de las señales en tiempo discreto es generalmente una de las herramientas más utilizadas en el procesamiento digital de señales.
Para realizar el análisis en frecuencia sobre una señal en tiempo discreto *x(n)*, se convierte la secuencia del dominio del tiempo a su equivalente en el dominio de la frecuencia, tal representación en la frecuencia está dada por la trasformada de Fourier *X(w)* de la secuencia *x(n)*, sin embargo *X(w)* es una función continua en frecuencia y por lo tanto no es computacionalmente eficiente [1].

Es una representación de la secuencia *x(n)* a partir de muestras de su espectro *X(w)*, tal representación en el dominio de la frecuencia es la trasformada de Fourier en tiempo discreto, el muestreo de *X(w)* es realizado por N muestras espaciadas $w_k=2\pi k/N$, $k=0,1,2,...,N-1$. Por lo tanto la trasformada discreta de Fourier queda definida como:

$$x(k)=\sum_{n=0}^{N-1}x(n)W_N^{Kn} \qquad k=0,1,...,N-1 \qquad 2.15$$

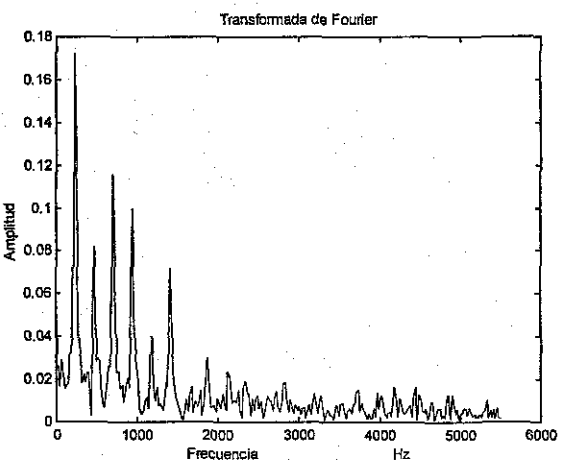
$$W_n=e^{-j2\pi/N}$$

La transformada inversa queda definida por:

$$x(n)=\frac{1}{N}\sum_{n=0}^{N-1}x(k)W_N^{-Kn} \qquad n=0,1,...,N-1 \qquad 2.16$$

$$W_n=e^{-j2\pi/N}$$

La Gráfica 2.7 muestra la trasformada de Fourier para un marco sonoro de la señal de voz.



Gráfica 2.7 Transformada de Fourier

Capítulo 3

Predicción Lineal

Introducción

La cuantización vectorial (VQ por sus siglas en inglés) es un principio de compresión de datos, con diferentes áreas de aplicación como codificación de voz, codificación de imágenes y reconocimiento de voz. La cuantización vectorial es una técnica para la compresión de la información. Por ejemplo para una tasa de transmisión especifica el objetivo de la cuantización vectorial es encontrar un conjunto de vectores de reproducción o code book, que representen la información con una distorsión mínima. La codificación de voz por VQ es una técnica basada en el Código de Predicción Lineal(LPC por sus siglas en inglés).

Esta es una de las técnicas más usadas en el procesamiento de señales de voz además, ha probado ser muy eficiente debido a la posibilidad de parametrizar la señal con un número pequeño de características con las cuales es posible reconstruirla. Los parámetros obtenidos mediante este método se caracterizan por variar en forma lenta durante las ventanas de tiempo de análisis. Mediante ella podemos representar a la señal de voz por parámetros que varían en el tiempo los cuales están relacionados con la función de transferencia del tracto vocal y las características de la fuente.

3.1 Análisis de Predicción Lineal

Resultados experimentales han mostrado que existe una buena correlación entre muestras adyacentes de voz, tomando en cuenta el resultado de estos experimentos se puede escribir un predictor de la siguiente manera [3]:

$$S_n \cong a_1 S_{n-1} + a_2 S_{n-2} + \ldots + a_p S_{n-p} \qquad 3.1$$

Esta relación asume que el valor de una muestra de voz es predecible por la sumatoria de las *p* muestras pasadas, las cuales son multiplicadas por el factor de peso *ai*. A estos valores se les llama coeficientes de predicción lineal.

El calculo de la muestra S_n, no requiere demasiado tiempo de procesamiento, lo cual es importante en la implementación. El modelo matemático expuesto establece que el tracto vocal puede modelarse mediante un filtro digital siendo los parámetros los que determinan la función de transferencia. El modelo consiste en dado un segmento de palabra, extraer los coeficientes del filtro. El análisis de predicción lineal permite aproximar una señal a partir de muestras pasadas. En este caso se trata de predecir señales de voz mediante un filtro FIR (filtro de respuesta impulsiva finita), cuya función de transferencia se deduce de:

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + Gu(n) \qquad 3.2$$

La señal de voz se representa por medio de señales anteriores y *u(n)* viene a ser la entrada del filtro. La entrada al filtro será un tren de impulsos periódicos o una fuente de ruido aleatorio. El tren de impulsos producirá señales sonoras mientras que la fuente de ruido aleatorio producirá señales no sonoras a la salida del filtro.

De esta manera el filtro viene a representar un modelo del tracto vocal. La función de transferencia del filtro se obtiene sacando la transformada **Z**.

$$H(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-1}} \qquad 3.3$$

Donde G es la ganancia del filtro y dependerá de la naturaleza de la señal *s(n)*. El problema consiste en determinar los coeficientes de predicción a_k y la ganancia. Los parámetros de predicción lineal pueden ser utilizados para realizar síntesis o reconocimiento de voz. El cálculo se realiza minimizando el error que se comete cuando se intenta realizar la aproximación de la señal.

Sea $\hat{\boldsymbol{s}}_p$ la señal estimada a partir de la señal **s** original, entonces:

$$\hat{s}(n) = -\sum_{k=1}^{p} a_k \hat{s}(n-k) \qquad 3.4$$

El error entre la señal real y la señal de predicción será

$$e(n) = s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^{p} a_k \hat{s}(n-k) \qquad 3.5$$

Para calcular los coeficientes a_k se utiliza la expresión en la cual el error medio cuadrático es minimizado sobre todas las muestras disponibles. Los coeficientes de predicción se calculan mediante el método de los mínimos cuadrados minimizando el error cuadrático medio con respecto a cada uno de los coeficientes a_k. Con lo cual se obtiene el siguiente conjunto de ecuaciones:

Se realiza la minimización con respecto a $\boldsymbol{a}_k$

$$\sum_{k=1}^{p} a \sum_{n} \hat{s}(n-k)\hat{s}(n-1) = \sum_{n} \hat{s}(n)\hat{s}(n-1) \qquad 3.6$$

$$\begin{aligned} &a_1 r(0)+a_2 r(1)+\ldots+a_p r(p-1)=-r(1) \\ &a_1 r(1)+a_2 r(0)+\ldots+a_p r(p-2)=-r(2) \\ &\vdots \\ &a_1 r(p)+a_2 r(p-1)+\ldots+a_p r(0)=-r(p) \end{aligned} \qquad 3.7$$

En forma matricial

R.a=-r

Donde:
r^T=[r(1) r(2) ...r(p)]
a^t=[a(1) a(2) ... a(p)]

$$R = \begin{bmatrix} r(0) & r(1) & r(2) & r(3) & \cdots & r(p-1) \\ r(1) & r(0) & r(1) & r(2) & & r(p-2) \\ r(2) & r(1) & r(0) & r(1) & & r(p-3) \\ r(p-1) & r(p-2) & r(p-3) & r(0) & & r(0) \end{bmatrix} \qquad 3.8$$

Donde r(i) se obtiene de

$$r(i) = r(-i) = \sum_{n=0}^{N-i-1} s(n)s(n+i) \qquad 3.9$$

La ecuación anterior es la función de autocorrelación la cual proporciona una medida de la correlación de la señal con una copia defasada en el tiempo de sí misma. De aquí se extraen los p coeficientes de autocorrelación. Los valores típicos de p pueden estar entre 10 y 15 [10].

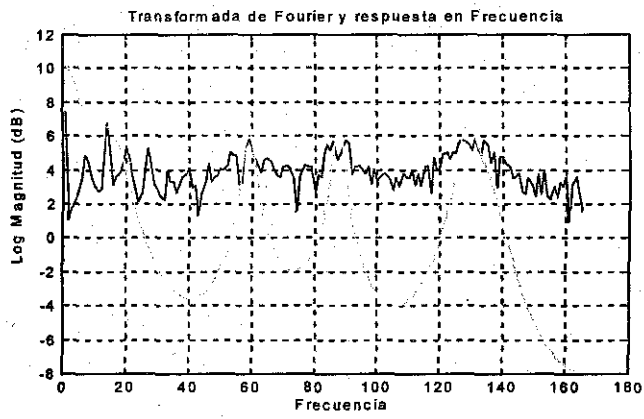
El método anterior para el cálculo de las a_i es llamado método de autocorrelación, a la matriz R se le conoce como matriz de Teopliz. Una matriz de Teopliz es aquella cuyas diagonales tienen los mismos elementos además es una matriz no singular lo que implica que tiene inversa y tiene la siguiente solución.

$A=-R^{-1}r$

Para resolver esta ecuación se recurre al algoritmo de Levinson-Durbin el cual permite resolver el sistema de ecuaciones de una forma eficiente:

$$\begin{aligned} &E(0) = r(0) \\ &K_i = \frac{r(i) + a_1^{i-1} r(i-) + \ldots + a_{1i-1}^{i-1} r(1)}{E(i-1)} \qquad para \qquad i = 1,\ldots p \\ &a_i^i = k_i \\ &a_j^i = a_j^{i-1} + k_i a_{i-j}^{i-1} \qquad j = 1,\ldots, i-1 \\ &E(i) = (1 - K_i^2) E(i-1) \end{aligned} \qquad 3.10$$

Dados los coeficientes del filtro a_k se tiene para la ventana de análisis, la función de transferencia del modelo del tracto vocal en ese instante es decir, se tiene la forma del comportamiento de la cavidad vocal y con la señal de excitación se obtiene el sonido emitido en ese momento. Para comprobar este hecho podemos comparar el espectro LPC alcanzado con el espectro de la señal obtenida mediante la transformada discreta de Fourier. La Gráfica 3.1 muestra una porción de señal de voz en donde se puede observar que el espectro LPC, que corresponde a la curva suave, en cierta forma envuelve al espectro de la señal de voz.



Gráfica 3.1 Respuesta en frecuencia de los coeficientes LPC.

3.2. Cuantización Vectorial

La cuantización vectorial (VQ) es un principio de compresión de datos con diferentes áreas de aplicación como: codificación de voz, codificación de imágenes y reconocimiento de voz. La cuantización vectorial es una técnica para la compresión de la información. Por ejemplo para un tasa de transmisión especifica el objetivo de la cuantización vectorial es encontrar un conjunto de vectores de reproducción o code book, que representen la información con una distorsión mínima. La codificación de voz por VQ es una técnica basada en el Código de Predicción Lineal(LPC) [13].

Si se asume que $x=[x_1\ x_2\ x_3...x_N\ ]$ es un vector N-dimensional cuyos componentes $\{\ x_1\ x_2\ x_3...x_N\ \}$ son variables aleatorias reales, en cuantización vectorial, si el vector x es mapeado a otro vector también real, se dice que x está cuantizado como y, donde y es el valor cuantizado de x, por lo tanto podemos escribir:

$y=q(X)$.

Donde $q()$ es el operador de cuantización. También a y se le denomina vector de reconstrucción o vector de salida que corresponde a x. Típicamente y, toma un conjunto finito de valores $Y=\{y_i,\ 1< i <L\}$, donde $Y_i=[y_1\ y_2\ ...y_N]$. Al conjunto Y se le conoce como el diccionario de reconstrucción o code book, L es el tamaño del diccionario y $\{y_i\}$ es el conjunto de vectores de códigos. Los vectores y_i también son conocidos en la literatura de reconocimientos de patrones como los patrones de referencia. El tamaño L del diccionario también se conoce como número de niveles, expresión utilizada en la terminología de la cuantización escalar. Entonces se puede hablar acerca de un diccionario de L niveles. Para el diseño de éste, se particiona el espacio N dimensional del vector aleatorio X en L regiones o celdas $\{c_i,\ 1<y<L\}$ y se asocia c_i a un vector y_i . El cuantizador entonces asigna el vector de código y_i si x está en c_i [12].

$Q(x)=y_i$ si $x \varepsilon c_i$

Al proceso de diseño del diccionario también se le conoce como entrenamiento o población del diccionario. La Figura 3.1 muestra un ejemplo de particionamiento en un espacio bidimensional *(N=2)* como demostración de la cuantización vectorial. La región limitada por las líneas oscuras corresponde a la celda c_i.

Cualquier vector *x* que cae en ésta celda es cuantizado como y_i. Las posiciones de otros vectores de código que corresponden a otras celdas se muestran como puntos. El número total de vectores de código en éste ejemplo es *L = 18* [12].

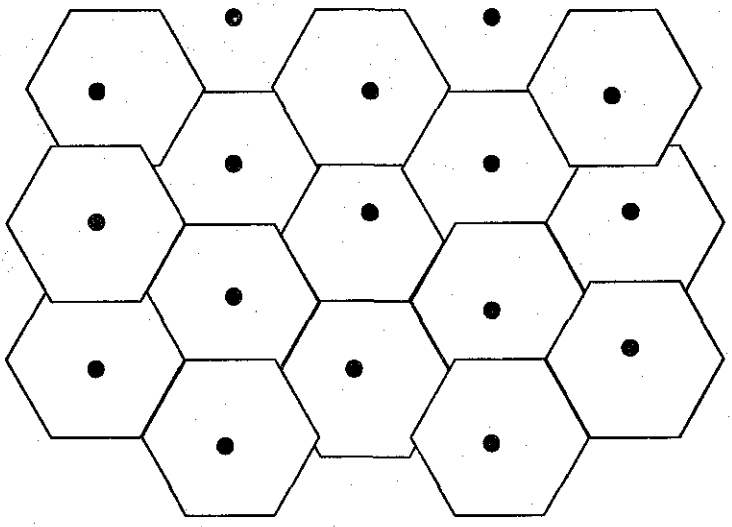


Figura 3.1 Partición de un espacio bidimensional N=2 y L=18 grupos

3.3 Medidas de Distorsión

Para que sea de utilidad, una medida de distorsión o distancia debe ser perceptible, de tal forma que pueda ser analizada y calculada además debe ser relevante en forma subjetiva, de modo que las diferencias en los valores de la distorsión puedan ser utilizados para indicar diferencias de similitud en la calidad de voz. La mayoría de las medidas de distorsión en uso actual son perceptuales y en cierta medida relevantes subjetivamente. Se ha descubierto que si la distorsión disminuye en pocos decibeles, en algunas situaciones será perceptible y en otras no.

Mientras que las medidas de distorsión objetivas son herramientas necesarias y útiles en el diseño de sistemas de codificación, se requiere hacer ***n*** pruebas de calidad subjetiva para mejorar el desempeño del sistema.

Una medida de distancia debe obedecer a las siguientes propiedades [1]:

1) No Negatividad

$$0 \le d(x,y) < \infty \qquad para \quad todo \quad x,y \in X$$

$$d(x,y) = 0 \Leftrightarrow x = y$$

2) Simetría

$$d(x, y) = d(y, x)$$

3) Desigualdad del triángulo

$$d(x,y) = d(x,z) + d(y,z)$$

A continuación se enumeran algunas de las medidas de distorsión más utilizadas.

3.4 Error Cuadrático Medio (MSE).

Es la medida de distorsión más utilizada, donde la distorsión está definida como:

$$d_r(\underline{x},\underline{y}) = \frac{1}{N}\sum_{k=1}^{N} | x_k - y_k |^r \qquad 3.11$$

La popularidad de MSE se basa en sus simplicidad y su seguimiento matemático. Se puede definir una distorsión más general basada en la normal L_r como:

$$d(\underline{x},\underline{y}) = \frac{1}{N}(x-y)^T(x-y) = \frac{1}{N}\sum_{k=1}^{N}(x_k - y_k)^2 \qquad 312$$

3.5 Medidas de Distorsión de Predicción Lineal

En el análisis LPC, los coeficientes de predicción *{a(k)}* se obtienen como resultado de la minimización de energía del residuo de la predicción. Se puede demostrar que la solución para valores de A(z) óptimos es única y se calcula en función del conjunto de ecuaciones lineales simultáneas [2]

$$\sum a(k)r(i-k) = -r(i) \qquad 1 \le i \le N \qquad 3.13$$

Donde *r(i)* se refiere a los coeficientes de autocorrelación de tiempo corto de la señal de voz sobre una trama. La ganancia *G* del filtro *H(z)* se calcula de forma que cuando es excitado por una fuente de variancia unitaria la energía de salida será igual a *r(0)*.

Los parámetros de filtro *H(z)* se pueden calcular para cada trama, se cuantizan y se transmiten. La ganancia G normalmente se cuantiza en una escala logarítmica y se transmite por separado. Debido a que la cuantización de los coeficientes de predicción pueden generar inestabilidad del filtro resultante, normalmente se transforman a otro conjunto de parámetros conocidos como los coeficientes de reflexión *{k_k,1<k<N}* o coeficientes de correlación parcial.

$$s_k = \frac{2}{\pi}\sin^{-1} k_k \qquad 1 \le k \le N$$

$$G_k = \frac{1}{2}\log\frac{1+k_k}{1-k_k} = \tanh^{-1} k_k \qquad 1 \le k \le N \qquad 3.14$$

Los parámetros G_k son conocidos como las razones logarítmicas del área (LARs) de la analogía del tubo acústico del tracto vocal, y poseen la propiedad de que pequeños cambios en G_k son aproximadamente proporcionales en el espectro logarítmico de $H(z)$.

3.6 Medida de Distorsión de Itakura-Saito

Una medida de distorsión alternativa utilizada en la cuantización, son los coeficientes de predicción propuestos por Itakura y Saito que se derivan de los principios de máxima similitud [10].

Una de las primeras medidas de distancia introducidas al reconocimiento de voz se basó en el error de predicción mínimo y principios de agrupamiento espectral, a ésta se le conoce como medición de máxima similitud. Esta medición calcula la energía de la diferencia en el espectro de dos conjuntos de parámetros LPC. Esencialmente, evalúa la similitud de los datos de prueba que han sido generados por un modelo estadístico basado en conjunto de parámetros LPC de referencia. Esta medida de distancia está dada por:

$$D = (a, \hat{a}) = \frac{a^T R a}{\hat{a} R a^{`}} \qquad 3.15$$

en donde R representa la matriz de autocorrelación usada para generar los parámetros .

Generalmente en la expresión anterior es necesario realizar el cálculo con matrices, para realizar estas operaciones existe un método más eficiente el cual utiliza las secuencias de autocorrelación $r(i)$ de la señal de voz y la secuencia de autocorrelación de los coeficientes del filtro $r_a(i)$.

La secuencia de autocorrelación de los filtros queda definida como:

$$r_a(i) = \sum_{k=i}^{p} a_k a_{k-i} \qquad i = 0,\ldots, p \qquad 3.16$$

entonces $a^T R a$ se calcula como:

$$\mathrm{a^T Ra} = \mathrm{r_a}(0)\mathrm{r}(0) + 2\sum_{k=1}^{\mathrm{p}} \mathrm{r_a}(i) r(i) \qquad 3.17$$

A pesar que el cálculo en (3.15) implica la multiplicación de una matriz, el cálculo puede simplificarse considerablemente y se reduce a un producto escalar (punto) con esta nueva expresión.

3.7 Diseño del Diccionario

Para diseñar un diccionario de L niveles, particionamos el espacio N dimensional en L celdas $\{c_i\ 1<i<L\}$ y asociamos cada celda c_i con un vector y_i. El cuantizador entonces asigna el vector de código y_i, si x está en c_i. Un cuantizador se dice que es óptimo (de distorsión mínima) si la distorsión es minimizada sobre todos los cuantizadores de L niveles. Existen dos condiciones para que un cuantizador sea óptimo. La primera condición es que el cuantizador óptimo se realice usando una regla de distorsión mínima o del vecino más cercano.

$Q(x)=y_i$ *si y solo si* $d(x,y_i)$, *j diferente de i* $1<j<L$

Esto es, el cuantizador toma el vector del código que resulta de la distorsión mínima con respecto a *x*. La segunda condición necesaria para que sea óptimo, es que cada vector de código *y* se tome para minimizar la distorsión promedio en la celda c_i. Esto es:

$$D_i = \left[d(x,y)\middle| x \in C_i\right] = \int_{x \in c_i} d(x,y)\,p(x)\,dx \qquad 3.18$$

A tal vector lo denominaremos centroide de la celda c_i, y escribimos

$y_i=cent(C_i)$

El cálculo del centroide para una región particular dependerá de la definición de la medida de distorsión. Las celdas definidas se conocen como celdas del vecino más cercano o regiones de Dirichlet. En la práctica obtenemos una serie de vectores de entrenamiento $\{x(n),\ 1<n<M\}$. Un subconjunto de M_i de estos vectores se encontrará asignado a la celda C_i. La distorsión promedio *D* está dada por

$$D_i = \frac{1}{M}\sum_{x \in C_i} d(x,y) \qquad 3.19$$

Para los criterios MSE o MSE ponderado, se puede demostrar que *D* es minimizada por

$$y_i = \frac{1}{M}\sum_{x \in C_i} x(n) \qquad 3.20$$

Simplemente y_i es la media muestral de todos los vectores de entrenamiento contenidos en C_i.

Un método para el diseño de diccionarios es un algoritmo iterativo de agrupamiento conocido en la literatura de reconocimiento de patrones como algoritmo de K-Medias. Para nuestro caso *k=L*. El algoritmo divide el conjunto de vectores de entrenamiento $\{x(n)\}$ en L grupos C_i de tal forma que las condiciones de optimización se cumplan. Después, existe un índice de iteración *m* y $C_i(m)$ es el i-ésimo grupo en la iteración *m*, con $y_i(m)$ como su centroide.

3.8 Algoritmo de K-Medias

A continuación se describe el algoritmo de K-Medias también llamado algoritmo de Lloyd generalizado.

Paso 1. Inicialización: asignar $m= 0$. Escoger un método adecuado para determinar un conjunto inicial de vectores de código $y_i\ (0),\ 1<i<L$.

Paso 2. Clasificación: clasificar el conjunto de vectores de entrenamiento $\{x(n),\ 1 < n < M\}$ a grupos de C_i mediante la regla del vecino más cercano.

$$x \in C_i(m) \Leftrightarrow d[x, y_i(m)] \leq d[x, y_j(m)] para \quad toda \quad j \neq i \qquad 3.21$$

Paso 3. Actualización de los vectores de código: *m...m +1*. Actualizar el vector de código de cada grupo, calculando el centroide de los vectores de entrenamiento en cada grupo.

$$y_i(m) = cent(C_i(m)) \qquad 1 \leq i \leq L$$

Paso 4. Prueba de terminación: si el decremento en la distorsión total *D(m)* en la iteración *m* relativa a *D(m-1)* está por debajo del umbral, se detiene; en caso contrario, volver al Paso 2.

El algoritmo mencionado converge a un óptimo local, pero cualquier solución en general no es única. La optimización global se puede obtener mediante la inicialización de los vectores de código a diferentes valores repitiendo el algoritmo para diferentes valores de inicialización y escogiendo el diccionario que resulte en la menor distorsión de todos.

3.9 Detección del Tono utilizando la Función de Autocorrelación

Un método para la localización del tono es el método de autocorrelación, el cual sirve para localizar el periodo de una señal, dicha propiedad sirve como herramienta para localizar el periodo del tono de la señal de voz de un parlante. Una de las dificultades para lograr una estimación confiable del tono a través de un amplio rango de expresiones y parlantes, es la estructura Formant sobre las mediciones relativas a la periodicidad de la señal voz [2]. Por tanto para lograr una detección confiable del tono, es deseable que los efectos de las resonancias del conducto vocal sobre la señal de voz sean reducidos al máximo.

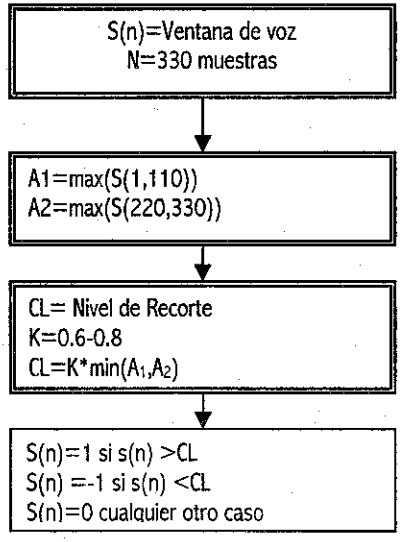
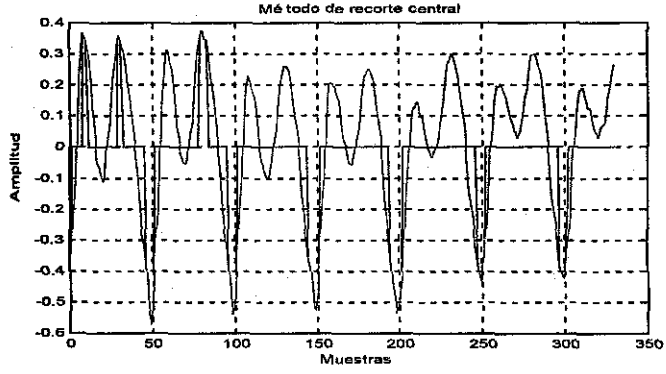


Diagrama 3.1 Método de recorte central

El método de autocorrelación propone un recorte central de la señal voz para reducir los efectos de las resonancias del conducto vocal sobre la periodicidad de la señal. Para localizar el tono se utiliza el método llamado recorte central que consiste en suprimir la señal entre ciertos niveles. Este método se describe en el Diagrama 3.1.

Debido al amplio rango dinámico de la señal de voz, el nivel de recorte se escoge de tal manera que se prevengan pérdidas de información, sobre todo en los frames de sonidos sonoros, en los que la señal está incrementando o decrementando su amplitud. El modo en el cual el nivel de recorte se fija, es el siguiente: el frame de voz el cual consta de *N* muestras, se divide en tres partes iguales, a continuación el algoritmo hallará el máximo absoluto tanto de la primera parte como de la tercera de tal manera que se tendrán dos máximos [10]. Entonces el nivel de recorte se fija como un porcentaje del menor de estos dos máximos. Extensivas simulaciones por computadora han demostrado que dicho porcentaje es del 80 % para la mayoría de los casos. El efecto del método de recorte central es deshacer la información de los formantes y retener la periodicidad de la señal voz. La Gráfica 3.2 muestra como se modifica la señal con este método.



Gráfica 3.2 Método de recorte central

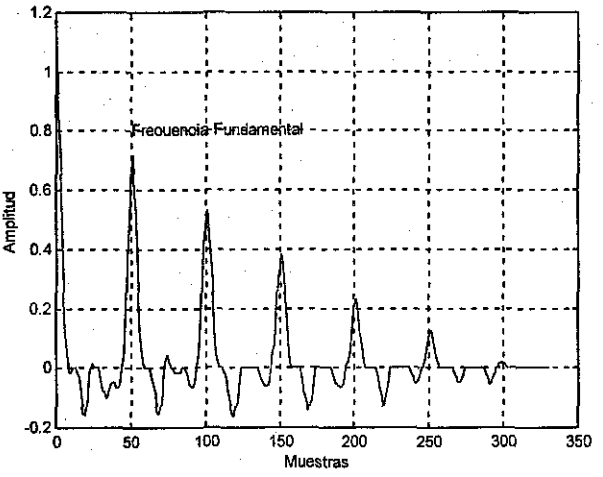
El siguiente paso del algoritmo es el calculo de la función de autocorrelación para el marco de voz:

$$R_x(m) = \sum x(n)x(n+m) \qquad m = M_i, M_i + 1, \ldots, M_f \qquad 3.21$$

Donde M_i es el retraso inicial y M_f es el retraso final, para los cuales la función de autocorrelación es calculada. Los valores típicos para M_i y M_f son 25 y 200 respectivamente, los cuales corresponden a un rango de tono de 25 a 200 Hz, para una frecuencia de muestreo de 10000 Hz. Adicionalmente $R_x(0)$ es calculada con el objeto de normalizar la función de autocorrelación.

Se debe hacer notar que en el cálculo de la función de autocorrelación, se asume que las muestras que se encuentra fuera del frame son cero, esto implica que la función de autocorrelación sufre de una ponderación lineal, la cual es uno para $m = 0$ y decae linealmente hasta llegar a cero para $m = 256$. Está ponderación lineal tiene el efecto de realzar el pico correspondiente al periodo tono, con respecto a los picos que corresponden a múltiplos de dicho periodo, reduciendo de esta manera la posibilidad de que el tono encontrado se duplique o se triplique.

El último paso en la estimación del tono consiste en hallar el valor máximo de la función de autocorrelación normalizada con respecto a $R_x(0)$ en el intervalo M_l y M_f. Tanto la localización como el valor pico máximo son guardados en memoria. Si el valor máximo excede un umbral sonoro no-sonoro, en el orden de 0.3, el frame de voz es clasificado como sonoro y el periodo tono es igual a la posición del pico máximo, si el pico máximo cae por debajo del valor de dicho umbral, el frame de voz es declarado como no sonoro. La Gráfica 3.3 muestra como el método de autocorrelación muestra la periodicidad de la señal, además el pico máximo indica la posición del tono, el cual puede ser utilizado para el reconocimiento del parlante.



Gráfica 3.3. Periodicidad de la señal

3.10 Análisis Cepstral

De acuerdo a nuestro modelo usual, el habla está compuesta de una secuencia de excitación convolucionada con la respuesta impulso del modelo vocal. Generalmente, solo se tiene la salida, sin embargo, a menudo encontramos deseos de eliminar uno de los componentes para que el otro pueda ser examinado, codificado y modificado usando un algoritmo de reconocimiento.

La Transformada de Fourier se puede aplicar en señales hechas bajo una combinación lineal, por ejemplo cuando una señal de frecuencia $x_1(n)$, es corrompida por la adición de un ruido $w(n)$, como se muestra en la siguiente ecuación:

$$X(n)=x_1(n)+w(n) \qquad 3.22$$

Debido a que la transformada de Fourier es un operador lineal sabemos que un espectro de magnitud de $X(n)$ nos permitirá examinar la secuencia de los componentes. El espectro es la representación de la señal con la cual podemos apreciar su "separación". Además las representaciones de los componentes de la señal están combinadas linealmente en el espectro.

Si el objetivo fuera apreciar algunas propiedades del ruido, por ejemplo, podríamos ser capaces de derivar la información deseada del espectro. Sin embargo si el objetivo fuera remover el ruido de la señal podríamos presumiblemente diseñar un filtro paso bajas para remover la indeseable componente de alta frecuencia. Desde el punto de vista del dominio de la frecuencia consistiría en la construcción de un sistema que removería el espectro de energía de alta frecuencia indeseada. Los resultados podrían ser transformados de regreso al interior del dominio del tiempo.

Cada operación tomada para producir este filtrado resulta ser lineal, por lo tanto la operación completa se dice que es lineal. Solamente porque $x_1(n)$ y $w(n)$ están combinadas linealmente podemos confiar que poniendo la señal $x(n)$ dentro del filtro producirá solamente la parte de baja frecuencia $x_1(n)$ lo cual quedaría como.

$$F(x(n))=F\{x_1(n)+w(n)\}= F(x_1(n))+ F(w(n)) \qquad 3.23$$

Si los componentes fueran combinados en alguna otra forma, por ejemplo convolución, no podríamos tener una idea clara de los efectos del filtro en las partes de los componentes.

$$F(x(n))=F\{x_1(n)*w(n)\} \qquad 3.24$$

Sobre esta situación es el caso con el problema del la señal de voz a la cual nos dirigimos. El Análisis "cepstral" es motivado y diseñado para tratar asuntos relacionados con la voz del hablante [4]. De acuerdo a nuestra modelo de producción del habla, la voz es compuesta de la convolución de la secuencia de excitación con la respuesta impulso del canal vocal.

$$S(n)=e(n)*x(n) \qquad 3.25$$

Debido a que las partes individuales no son combinadas, las acostumbradas técnicas lineales no parecen proveer de ayuda. Como el espectro, el cepstrum representará una transformación en la señal del habla con dos importantes propiedades:

- Serán separados en el cepstrum.
- Serán linealmente combinadas en el cepstrum.

Si este es le propósito de apreciar algunas propiedades de las señales, el cepstrum por sí mismo puede ser suficiente para proveer la información necesaria. Si éste es el propósito de eliminar uno de los componentes de la señal. Sin embargo como los componentes representativos de la señal están linealmente combinados, estaremos habilitados para usar filtros lineales para separar componentes cepstrales indeseados.

Finalmente el cepstrum quedaría matemáticamente definido de la siguiente manera:

$$s(e^{j\theta}) = H(e^{j\theta})E(e^{j\theta}) \qquad 3.26$$

Donde:

$H(e^{j\theta})$ y $E(e^{j\theta})$

representa la envolvente espectral de la señal de voz y la excitación, para la representación en frecuencia se calcula la trasformada de Fourier de la señal de voz.

De las propiedades de los logaritmos:

$$z = x * y$$
$$\log z = \log x + \log y$$

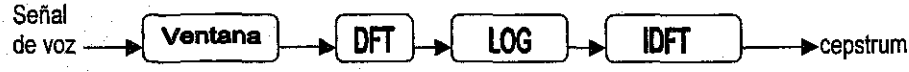
Aplicando esta propiedad a la ecuación 3.26

$$\log S(e^{j\theta}) = \log H(e^{j\theta}) + \log E(e^{j\theta}) \qquad 3.27$$

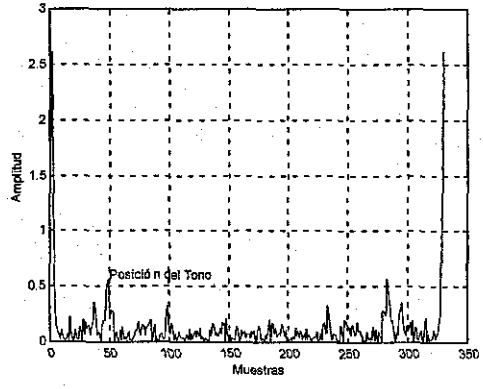
Para varias aplicaciones de procesamiento de voz solo se requiere la magnitud por lo tanto la ecuación anterior puede quedar escrita como:

$$\log S(|\ e^{j\theta}\ |) = \log H(|\ e^{j\theta}\ |) + \log E(|\ e^{j\theta}\ |) \qquad 3.28$$

El cálculo del cepstrum de realiza de la siguiente manera: a una ventana de voz se le aplica la trasformada de Fourier, al resultado de ésta se le aplica el logaritmo en ambos lados como en la ecuación 3.28. Finalmente se aplica la trasformada inversa de Fourier dando como resultado el cálculo del cepstrum. El diagrama del calculo del cepstrum queda definido como:



La Gráfica 3.4 muestra el cálculo del cepstrum real, el cual muestra la posición del tono para un sonido sonoro.



Gráfica 3.4 Cepstrum real

Capítulo 4

Redes Neuronales Artificiales

Introducción

Las Redes Neuronales Artificiales (en inglés Artificial Neural Networks ANNs) fueron originalmente una simulación abstracta de los sistemas nerviosos biológicos, formados por un conjunto de unidades llamadas "neuronas" o "nodos" conectadas unas con otras. Estas conexiones tienen una gran semejanza con las *dendritas* y los *axones* en los sistemas nerviosos biológicos.

Una primera clasificación de los modelos de ANN podría ser, atendiendo a su similitud con la realidad biológica:

- Los modelos de tipo biológico. Este comprende las redes que tratan de simular los sistemas neuronales biológicos así como las funciones auditivas o algunas funciones básicas de la visión o el habla.
- El modelo dirigido a aplicación. Estos modelos no tienen porque guardar similitud con los sistemas biológicos. Su arquitectura está fuertemente ligada a las necesidades de las aplicaciones para las que son diseñadas.

4.1 Redes Neuronales de Tipo Biológico

Se estima que el cerebro humano contiene más de cien mil millones (10^{11}) de neuronas y (10^{14}) **sinápsis** en el sistema nervioso humano. Estudios sobre la anatomía del cerebro humano concluyen que hay más de 1000 sinápsis a la entrada y a la salida de cada neurona. Es importante notar que aunque el tiempo de conmutación de la neurona (*unos pocos milisegundos*) es casi un millón de veces menor que en los actuales elementos de las computadoras, ellas tienen una conectividad miles de veces superior que las actuales supercomputadoras [7].

El objetivo principal de las redes neuronales de tipo biológico es desarrollar un elemento sintético para verificar las hipótesis que conciernen a los sistemas biológicos. Las neuronas y las conexiones entre ellas (sinápsis) constituyen la clave para el procesamiento de la información.
Observe la Figura 4.1

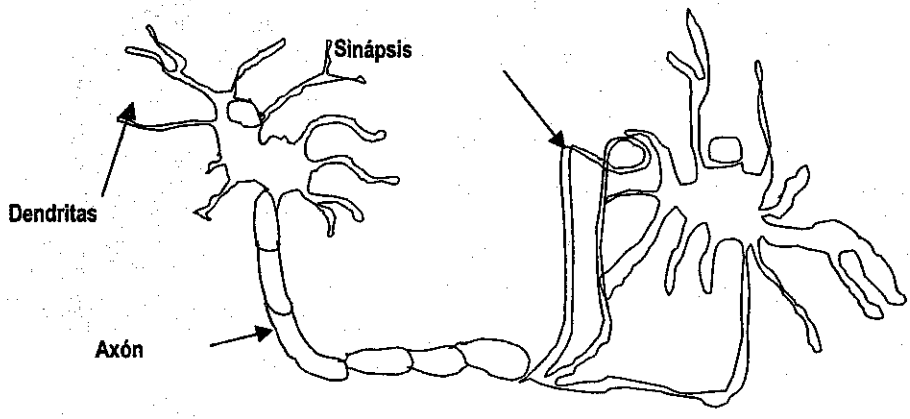


Figura 4.1 Neurona

La mayor parte de las neuronas poseen una estructura de árbol llamadas **dendritas** que reciben las señales de entrada que vienen de otras neuronas a través de las uniones llamadas **sinápsis**. Algunas neuronas se comunican solo con las cercanas, mientras que otras se conectan con miles. La neurona esta compuesta de:

- El cuerpo de la neurona
- Ramas de extensión llamadas **dendritas** para recibir las entradas
- Un **axón** que lleva la salida de la neurona a las dendritas de otras neuronas.

La forma que dos neuronas interactúan no está totalmente conocida, dependiendo además de cada neurona. En general, una neurona envía su salida a otras neuronas por su axón. El axón lleva la información por medio de diferencias de potencial u ondas de corriente, que dependen del potencial de la neurona. Este proceso es a menudo modelado como una regla de propagación representada por la función de red u(.) . La neurona recoge las señales por su sinápsis sumando todas las influencias excitadoras e inhibidoras. Si las influencias excitadoras positivas dominan, entonces la neurona da una señal positiva y manda este mensaje a otras neuronas por sus sinápsis de salida. En este sentido la neurona puede ser modelada como una simple función escalón *f(.)*. Como se muestra en la Figura 4.2, la neurona se activa si la fuerza combinada de la señal de entrada es superior a un cierto nivel, en el caso general el valor de activación de la neurona viene dado por una función de activación f(.).

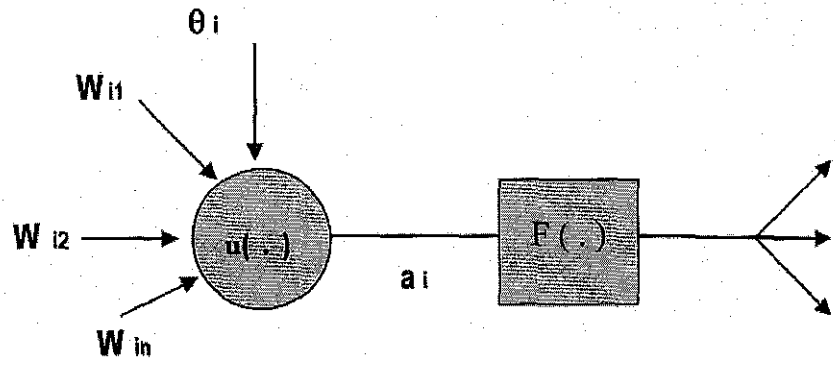


Figura 4.2 Bloques de una neurona

4.2 Red Neuronal Artificial

Una red neuronal es una red de muchos procesos simples cada uno de los cuales tiene memoria local, las unidades están interconectadas por canales de comunicación (conexiones) los cuales llevan un dato numérico a las unidades. Las unidades operan solamente en sus datos locales y en las entradas que reciben de las conexiones con algunas redes vecinas. Las ANN son modelos de redes neuronales biológicas. Muchas de las redes neuronales surgen del deseo de producir sistemas artificiales capaces de ser semejantes a la inteligencia o entendimiento humano [15].

Una definición interesante de red neuronal hace uso del concepto matemático de grafo, objeto consistente en un conjunto de nodos (o vértices), más un conjunto de conexiones establecidas entre ellos. En este caso, el grafo describe la arquitectura del sistema y proporciona los canales por los que puede describir su dinámica. Hay diferentes tipos de grafo, por ejemplo, grafo dirigido y no dirigido. En el primer tipo de grafo, las conexiones tienen asignado un sentido, mientras que en el otro son bidireccionales. Existen grafos densos cuando todos los nodos están conectados con casi todos y grafos dispersos cuando son pocas las conexiones entre los nodos. Un grafo puede componerse de diferentes tipos de nodos y diferentes tipos de conexiones [7].

Una forma de representar el grafo es, como su propio nombre indica, gráficamente, dibujando los nodos como círculos y las conexiones como líneas o flechas, según sean de un solo sentido o bidireccionales. Otra forma común de representación son mediante una matriz de conexiones. En el caso en que el grafo sea no dirigido, la matriz de conexiones será simétrica.

La definición matemática de red neuronal utilizando el concepto de grafo toma la siguiente forma:

1) A cada nodo *i* se asocia una variable de estado *Xi*
2) A cada conexión *(i,j)* de los nodos *i* y *j* de asocia un peso *wij*
3) A cada nodo *i* se asocia un umbral θ_i

4) Para cada nodo *i* se define una función $f(xj,wij, \theta j)$, que depende de los pesos de sus conexiones, del umbral y de los estados de los nodos *j* a él conectados. Esta función proporciona el nuevo estado del nodo.

En la terminología habitual de las redes neuronales, los nodos son las neuronas y las conexiones son las sinápsis.

4.3 Taxonomía de las Redes Neuronales

Existen dos fases en toda aplicación de las redes neuronales: la *fase de aprendizaje o entrenamiento* y la *fase de prueba*. En la fase de entrenamiento, se usa un conjunto de datos o patrones de entrenamiento para determinar los pesos (parámetros de diseño) que definen el modelo neuronal. Una vez entrenado este modelo, se usará en la llamada fase de prueba o funcionamiento directo, en la que se procesan los patrones de prueba que constituyen la entrada habitual de la red, analizándose de esta manera las prestaciones definitivas de la red.

- **Fase de Prueba**: los parámetros de diseño de la red neuronal se han obtenido a partir de unos patrones representativos de las entradas que se denominan *patrones de entrenamiento*. Los resultados pueden ser tanto calculados de una vez como adaptados iterativamente, según el tipo de red neuronal, y en función de las ecuaciones dinámicas de prueba. Una vez calculados los pesos de la red, los valores de las neuronas de la última capa se comparan con la salida deseada para determinar la validez del diseño.

- **Fase de Aprendizaje**: una característica de las redes neuronales es su capacidad de aprender. Aprenden por la actualización o cambio de los pesos sinápticos que caracterizan a las conexiones. Los pesos son adaptados de acuerdo a la información extraída de los patrones de entrenamiento nuevos que se van presentando. Normalmente, los pesos óptimos se obtienen optimizando (minimizando o maximizando) alguna "función de energía".

Las aplicaciones del mundo real deben acometer dos tipos diferentes de requisitos en el procesado. En un caso, se requiere la prueba en tiempo real pero el entrenamiento ha de realizarse "fuera de línea". En otras ocasiones, se requieren los dos procesos, el de prueba y el de entrenamiento en tiempo real. Estos dos requisitos implican velocidades de proceso muy diferentes, que afectan a los algoritmos y hardware usados. Atendiendo al tipo de entrenamiento las redes neuronales se clasifican como Supervisadas y No Supervisadas [15].

4.4 Neurona de una Sola Entrada

La Figura 4.3 muestra una neurona de una sola entrada, el escalar *p* (entrada) es multiplicado por un escalar de peso *w* para formar *w*p*, el cual es enviado al sumador, la otra entrada es multiplicada por un factor de bias *b* y después pasado al sumador. La salida del sumador *n*, frecuentemente referenciado como la entrada a la red ingresa a la función de trasferencia *f*, la cual produce una salida escalar ***a***.

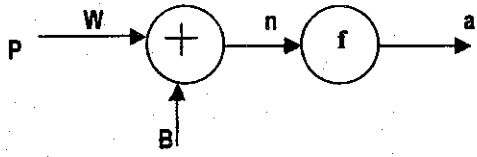


Figura 4.3 Neurona de solo una entrada

La salida escalar resultante es

$$a = f(wp+b)$$

Por ejemplo w=3, p=2 y b=-1.5

La salida actual depende en particular de la función de transferencia que se esté utilizando. Los parámetros *w* y *b* son parámetros escalares ajustables en la neurona. Típicamente la función de transferencia es seleccionada por el diseñador y los parámetros *w* y *b* son ajustados por alguna regla de aprendizaje de tal forma que la salida y entrada de la neurona busquen una meta especifica [7] [15].

4.5 Neurona de Múltiples Entradas

Generalmente una neurona tiene más de una entrada. La Figura 4.4 muestra una neurona con *r* entradas. Las entradas individuales p_1, p_2,...,p_n son multiplicadas por los correspondientes elementos $w_{1,1}$, $w_{1,2}$,...,$w_{1,r}$ de la matriz de pesos *W*.

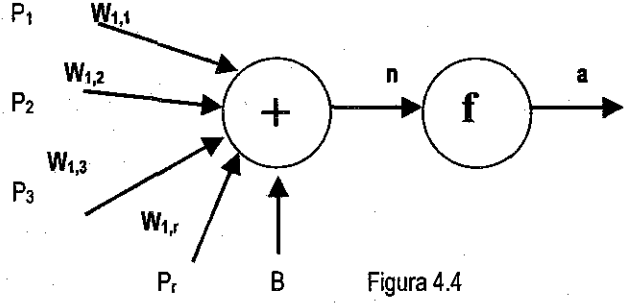


Figura 4.4

La neurona tiene un factor de bias, el cual es sumado al producto *wp* para formar la entrada a la neurona.

$$N = w_{1,1} p_1 + w_{1,2} p_2 + ... + w_{1,r} p_r + b \qquad 4.1$$

Donde la expresión anterior puede ser escrita en forma vectorial como:

$$N = WP + b \qquad 4.2$$

Entonces la salida de la neurona se puede escribir como:

$$a = f(WP + b) \qquad 4.3$$

4.6 Arquitectura de una Red Neuronal

Comúnmente una neurona con múltiples entradas puede no ser suficiente. Se pueden necesitar cinco o diez, operando en paralelo, en este caso se le llama arreglo. La Figura 4.5 muestra un arreglo de redes de *S* neuronas, note que cada una de las entradas *R* es conectada a cada neurona y la matriz de pesos tiene *S* renglones [15].

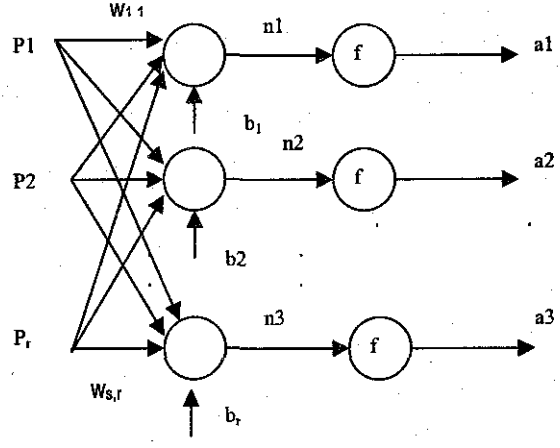


Figura 4.5 Arquitectura de una red neuronal

El arreglo incluye la matriz de pesos, el sumador, el vector de bias, la función de transferencia y el vector de salida *a*.

Cada elemento del vector de entrada *p* es conectado a cada neurona a través de la matriz de pesos *W*. Cada neurona tiene un factor de bias, un sumador, una función de trasferencia *f* y una salida *a*. Es común que el número de entradas a un arreglo sea diferente del número de neuronas. Además no todas las funciones de trasferencia del arreglo deben ser iguales.

La matriz de pesos queda definida como:

$$W = \begin{bmatrix} w1,1 & w1,2 \ldots & w1,r \\ w2,1 & w2,2, \ldots & w2,r \\ . & . & . \\ . & . & . \\ . & . & . \\ ws,r & ws,2 & ws,r \end{bmatrix} \qquad 4.4$$

Donde los índices de la matrices de los elementos de la matriz *W* indican la neurona destino asociada con su peso, mientras que el índice de la columna indica la entrada fuente, por ejemplo $w_{3,2}$ indica que la conexión es a la tercera neurona de la segunda entrada.

4.7 Múltiples Arreglos de Neuronas

Si ahora consideramos una red con múltiples arreglos, cada arreglo tiene su propia matriz de peso *W*, su propio vector de bias *b*, un vector de entrada a la red n y un vector de salida *a*. Se necesita introducir una nueva notación para distinguir entre los diferentes arreglos, se utilizará un índice para identificar cada uno de esos arreglos. Para denotar al vector de peso se utiliza *W1* y la matriz de peso del segundo arreglo puede ser escrito como *W2* y así sucesivamente [7].

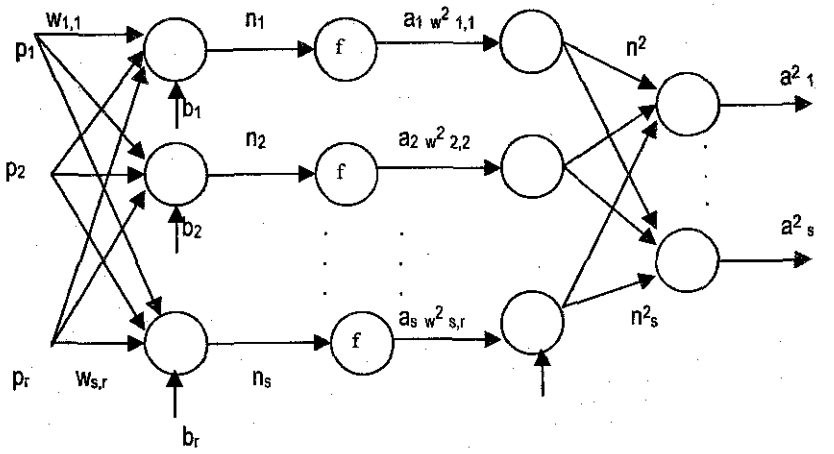


Figura 4.6. Arreglo de Neuronas.

La Figura 4.6 muestra un arreglo de neuronas en donde existen *r* entradas, *S1* neuronas en el primer arreglo, *S2* neuronas en el segundo arreglo, etc. Cada uno de los arreglos puede tener diferentes números de neuronas. Las salidas del arreglo uno son entradas al arreglo dos y las salidas del dos son entradas al tres. El arreglo cuya salida es la de la neurona, es llamado arreglo de salida, los otros arreglos son llamados arreglos ocultos. Neuronas de múltiples niveles o capas son más poderosas que las neuronas de un simple arreglo.

4.8 Aprendizaje

Aprendizaje es un proceso por el cual los parámetros libres de una red neuronal son adaptados a través de un proceso continuo de simulaciones en el ambiente en la cual se encuentra la red. El tipo de aprendizaje es determinado por la manera en la cual cambian los parámetros. La definición de proceso de aprendizaje implica la siguiente secuencia de eventos:

- La red neuronal es simulada por medio de un ambiente.

- Los cambios que sufre la neurona son resultado de la estimulación.
- La respuesta de la neurona es una nueva forma de ambiente, debido a los cambios internos que han ocurrido en su interior.

4.9 Reglas de Aprendizaje

Por reglas de aprendizaje se entiende el procedimiento para modificar los pesos y el factor de bias de una red neuronal, este procedimiento también es llamado algoritmo de entrenamiento. El propósito de las reglas de aprendizaje es entrenar a la red neuronal para realizar una determinada tarea. Existen varios tipos de reglas de aprendizaje para las redes neuronales, las cuales son supervisadas, y no supervisadas. La adaptación o aprendizaje es uno de los puntos focales de la investigación de redes neuronales.

4.9.1 Reglas de Entrenamiento Supervisado

Las redes de entrenamiento supervisado han sido los modelos de redes más desarrolladas desde inicios de las investigaciones de redes neuronales artificiales. Los datos para el entrenamiento están constituidos por varios pares de patrones de entrenamiento de entrada y de salida. El hecho de conocer la salida implica que el entrenamiento se beneficia por la supervisión de un maestro. Dado un nuevo patrón de entrenamiento. La Figura 4.7 muestra la forma de entrenamiento supervisado.

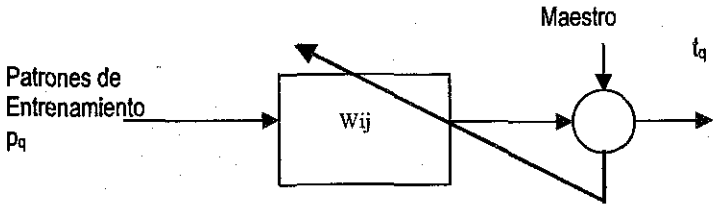


Figura 4.7. Entrenamiento Supervisado.

Donde p_q es una entrada a la red, t_q es la salida deseada. Como la entrada es aplicada a la red, la salida es comparada con la deseada. La regla de aprendizaje es usada para ajustar los pesos y el factor de bias de la red, de tal forma para acercar las salidas de la red con las deseadas. Las reglas del aprendizaje del perceptrón caen dentro de estas reglas supervisadas.

Las ANNs de entrenamiento supervisado constituyen la línea fundamental de desarrollo en este campo. Algunos ejemplos bien conocidos son la red perceptrón, ADALINE/MADALINE, y varias redes multicapa. En el entrenamiento supervisado hay dos fases a realizar: prueba y de entrenamiento.

En el entrenamiento supervisado, los patrones de entrenamiento se dan en forma de pares de entrada/maestro, donde M es el número de pares de entrenamiento. Dependiendo de la naturaleza de la información del maestro, hay dos aproximaciones al entrenamiento supervisado. Uno se basa en la corrección a partir de una decisión y la otra se basa en la optimización de un

criterio de costo. De la última, la aproximación del error cuadrático medio es el más importante. Las formulaciones de decisión y aproximación difieren en la información que tienen los maestros y la forma de usarla.

4.9.2 Reglas de Entrenamiento No Supervisado

Para los modelos de entrenamiento No Supervisado, el conjunto de datos de entrenamiento consiste sólo en los patrones de entrada. Por lo tanto, la red es entrenada sin el beneficio de un maestro. La red aprende a adaptarse basada en las experiencias recogidas de los patrones de entrenamiento anteriores. Este es un esquema típico de un sistema "No Supervisado". La Figura 4.8 muestra el comportamiento de una red neuronal no supervisada.

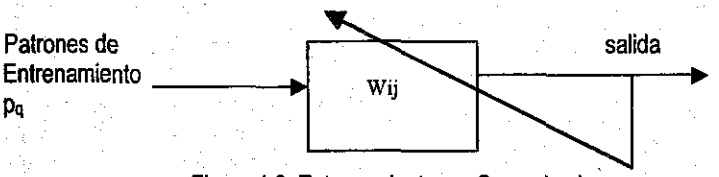


Figura 4.8. Entrenamiento no Supervisado.

Los pesos y el factor de bias son modificados en respuesta de las entradas a la red solamente, no existen patrones de salida disponibles, de primera vista este tipo de reglas parecen no ser prácticas. Entonces ¿cómo entrenar una red si no sabemos qué se quiere realizar?. Muchos de esos algoritmos realizan una operación de agrupamiento, estos aprenden de tal forma que pueden clasificar dentro de un número de clases finitas, este tipo de aprendizaje es especialmente útil en aplicaciones de cuantización vectorial.

En el aprendizaje competitivo como su nombre lo indica las salidas de las redes neuronales compiten entre ellas mismas para que solo una de ellas sea activada, este tipo de aprendizaje sirve para descubrir las características estadísticas, lo cual sirve para clasificar un conjunto de entradas en clases.

4.10 El Perceptrón

El Perceptrón fue el primer modelo de red neuronal desarrollado por Frank Rosenblatt en 1958 junto con otros investigadores. El perceptrón despertó un enorme interés en los años 60, debido a su capacidad para aprender a reconocer patrones sencillos. Sin embargo el perceptrón tenía limitantes ya que un perceptrón de un solo nivel solo puede ser utilizado para reconocer patrones que son linealmente separables. El perceptrón de un nivel puede usar entradas continuas y binarias. La Figura 4.9 muestra el perceptrón de 2 entradas con función de trasferencia *hardlim*.

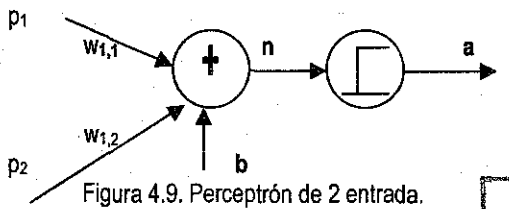


Figura 4.9. Perceptrón de 2 entrada.

El nodo calcula la suma de los pesos de los elementos de entrada, resta un umbral y pasa el resultado a través de un límite de no linealidad para que la salida sea *+1* o *–1*. La regla de decisión responderá clase A, si la salida es +1 y clase B, si la salida es –1. Una técnica útil para analizar el comportamiento de redes, tal como el perceptrón, es dibujar un mapa de regiones de decisión creando un espacio multidimensional, medido por la variables de entrada. Estas regiones de decisión especifican que valores de entrada pertenecen a la clase A y cuáles a la clase B. El perceptrón forma dos regiones de decisión separadas por un hiperplano. Cuando éstas son sólo 2 entradas y el hiperplano es una línea, las entradas arriba del límite de la línea pertenecen a la clase A y debajo de la línea a la clase B. Como puede verse en la Figura 4.10 la ecuación del límite de la línea depende de la conexión de los pesos y el umbral [15].

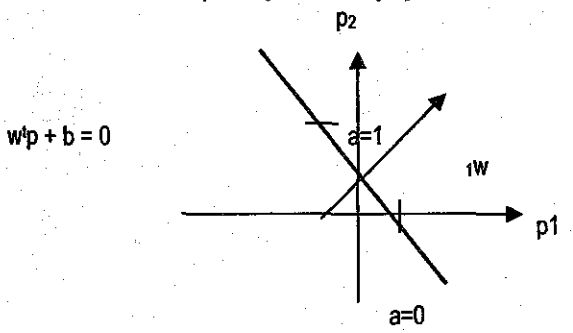


Figura 4.10 Límite de decisión de un perceptrón de 2 entradas

El algoritmo de aprendizaje del perceptrón es de tipo supervisado, lo cual requiere que sus resultados sean evaluados y se realicen las modificaciones necesarias al sistema para su actualización. Los valores de los pesos determina el funcionamiento de la red, estos valores pueden adaptarse utilizando diferentes algoritmos de entrenamiento.

El algoritmo de convergencia de ajuste de pesos para realizar el aprendizaje de un perceptrón con N entradas y un único elemento de salida queda descrito de la siguiente manera.

1.- ***Iniciar los pesos y el umbral.***

Inicialmente se asignan valores aleatorios a cada uno de los pesos W_i y al umbral - $W_{o=\text{alfa}}$.

2.- ***Presentación de un par de entradas y salidas esperadas:***

$p_n = (p_1, p_2, \ldots, p_n)$ junto con la salida deseada t_d

3. - ***Cálculo de la salida actual.***

$$a(t) = f\left(\sum W_{i(0)}(t) p_i(t) - \theta \right) \qquad 4.5$$

4.- ***Adaptación de los pesos:***

$$W_i(t+1) = W_i(t) + \alpha(t_d - a(t))P_i(t) \qquad 4.6$$

Donde t_d representa la salida deseada y será 1 si el patrón pertenece a la clase A y –1 si es de la clase B. En esta ecuación alfa es un factor de ganancia en el rango de 0.0 a 1.0. este factor debe ser ajustado de forma que satisfaga los requerimientos de aprendizaje rápido como la estabilidad de las estimaciones de los pesos. Este proceso se repite hasta que el error que se produce para cada uno de los patrones (diferencia entre el valor de salida y el obtenido es cero) o bien un valor preestablecido. Los pesos no cambian si la red ha tomado una decisión correcta.

5.- Volver al paso 2.

El perceptrón será capaz de aprender a clasificar todas sus entradas, en un número finito de pasos, siempre y cuando el conjunto de los patrones de entrada sea linealmente separable. En tal caso, puede demostrarse que el aprendizaje de la red se realiza en un número finito de pasos.

4.11 Perceptrón Multinivel

Un perceptrón multinivel o multicapa es una red de tipo feedforward compuesta de varias capas de neuronas entre la entrada y la salida. Esta red permite establecer regiones de decisión mucho más complejas que la de dos semiplanos, como lo hace el perceptrón de un solo nivel. La Figura 4.11 muestra un perceptrón multinivel con *k* capas ocultas [15].

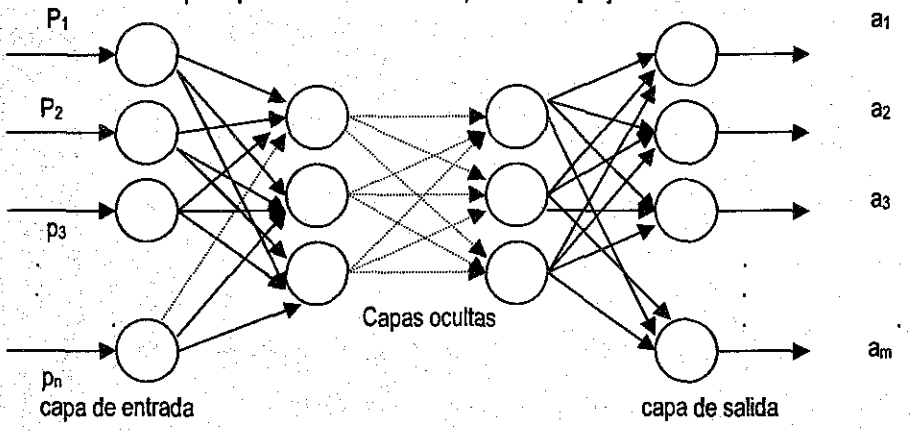


Figura 4.11 Perceptrón multinivel

El perceptrón con función de tipo escalón solo puede establecer dos regiones separadas por una frontera lineal. Un perceptrón con tres niveles de neuronas puede formar cualquier región convexa en el espacio. Las regiones convexas se forman mediante la intersección entre las regiones formadas por cada neurona de la segunda capa. Cada uno de estos elementos se comporta como un perceptrón simple, activando su salida para los patrones de un lado del hiperplano. Si el valor de los pesos de las conexiones entre las N_2 neuronas de la segunda capa y una neurona de nivel de salida son todos 1 y el umbral de la salida es N_2-a, donde $0<a<1$, entonces la salida de la red se activará sólo si las salidas de todos los nodos de la segunda capa están activos. Esto es equivalente a ejecutar la operación lógica AND en el nodo de salida, resultando una región de decisión que es la intersección de todos los semiplanos formados en el nivel anterior. La región de decisión resultante

de la intersección será una región convexa con un número de lados a lo sumo igual al número de neuronas de la segunda capa.

Este análisis introduce el problema de la selección del número de neuronas ocultas de un perceptrón de 3 capas. En general este número debe ser lo suficientemente grande como para formar una región compleja para la solución del problema, a la vez se debe considerar el inconveniente de un número grande de nodos.

Para entender mejor al perceptrón se da un ejemplo de la solución del problema de la XOR. Existen diferentes soluciones para este problema una de ellas es usar 2 neuronas en el primer nivel para crear dos regiones de decisión, la primera región separa p_1 de los demás patrones y la segunda región separa p_4. Entonces el segundo nivel es usado para combinar las 2 regiones usando la operación AND. Los límites de decisión son mostrados en la Figura 4.12.

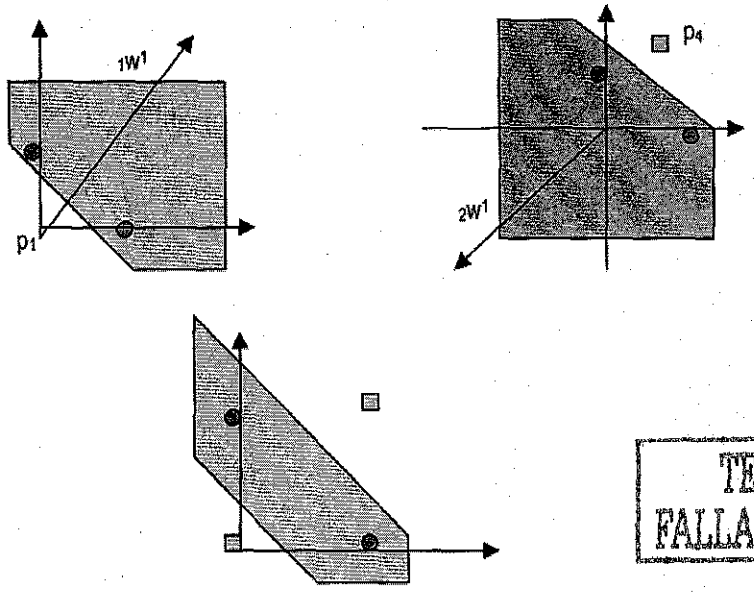


Figura 4.12 Límites de decisión

La región sombreada indica que las entradas producirán una salida de la red neuronal igual a 1.

4.12 Algoritmo de Backpropagation

El algoritmo de backpropagation para redes multicapa es una generalización del algoritmo LMS, ambos algoritmos calculan el error medio cuadrático. El algoritmo es entrenado con un conjunto de pares de entradas y salidas deseadas [15].

$$\{p_1,t_1\}, \{p_2,t_2\},\ldots, \{p_Q,t_Q\} \qquad 4.7$$

Donde p_q es una entrada a la red y t_q es la salida deseada. Conforme cada entrada es presentada a la red, la salida es comparada con las salidas deseadas. El algoritmo debe ajustar los parámetros de tal forma que se minimice el error medio cuadrático.

$$F(x) = E\left[e^2\right] = E\left[(t-a)^2\right]$$

4.8

Donde x representa el vector de pesos y bias, si la red tiene múltiples salidas esto se generaliza a:

$$F(x) = E\left[e^t e\right] = E\left[(t-a)^t (t-a)\right]$$

4.9

Como en el algoritmo LMS, se puede aproximar el error medio cuadrático por.

$$F(x) = \left[(t(k)-a(k))^t (t(k)-a(k))\right] = e^t(k)e(k)$$

4.10

Donde la esperanza matemática del error cuadrático ha sido remplazando por el error cuadrático en la iteración k. El algoritmo de los pasos descendientes para la aproximación del error medio cuadrático es:

$$W^m_{i,j}(k-1) = W^m_{i,j}(k) - \propto \frac{\partial \hat{F}}{\partial W^m_{i,j}}$$

4.11

$$b^m_i(k+1) = b^m_i(k) - \propto \frac{\partial \hat{F}}{\partial b^m_i}$$

4.12

Donde alfa es el ritmo de aprendizaje, este último desarrollo es igual al algoritmo LMS, la parte difícil es el cálculo de las derivadas parciales.

Debido a que el error es una función indirecta de los pesos en los niveles ocultos, se utiliza la regla de la cadena para calcular las derivadas, para revisar la regla de la cadena, se supone que se tiene una función f que es una función explícita solamente de la variable n, si se quiere realizar la derivada de f con respecto a la variable w entonces la regla de la cadena es:

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} x \frac{dn(w)}{dw}$$

4.13

Si utilizamos este principio para encontrar las derivadas de las ecuaciones 4.11 y 4.12 tenemos:

4.14

$$\frac{\partial \hat{F}}{\partial W^m_{i,j}} = \frac{\partial \hat{F}}{\partial n^m_i} x \frac{\partial n^m_i}{\partial W^m_{i,j}}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} x \frac{\partial n_i^m}{\partial b_i^m}$$

4.15

El segundo término en cada ecuación puede ser calculado fácilmente, debido a que la entrada al nivel *m* es una función explícita de los pesos, del factor de bias en ese nivel.

Por lo tanto

4.16

$$n_i^m = \sum_{j=1}^{s^{m-1}} W_{i,j}^m a_j^{m-1} + b_i^m$$

$$\frac{\partial n_i^m}{\partial W_{i,j}^m} = a_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1$$

4.17

Si se define

$$S_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}$$

4.18

La sensibilidad de *F* a los cambios en i-ésimo elemento de la entrada de la red en el nivel *m*, entonces las ecuaciones puede quedar definidas como

4.19

$$\frac{\partial \hat{F}}{\partial W_{i,j}^m} = S_i^m a_j^{m-1}$$

4.20

$$\frac{\partial \hat{F}}{\partial b_i^m} = S_i^m$$

Entonces podemos escribir el algoritmo de los pasos descendientes como:

4.21

$$W^m(k+1) = w^m(k) - \propto s^m (a^{m-1})^T$$

$$b^m(k+1) = b^m(k) - \propto S^m$$

4.22

Donde S^m se puede calcular de la siguiente forma:

$$S^m = \frac{\partial \hat{F}}{\partial n^m}\begin{bmatrix} \frac{\partial \hat{F}}{\partial n^m} \\ \frac{\partial \hat{F}}{\partial n^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n^m} \end{bmatrix}$$

4.23

$$S^m = \frac{\partial \hat{F}}{\partial n^m} = \left(\frac{\partial n^{m+1}}{\partial n^m}\right)^T \frac{\partial \hat{F}}{\partial n^{m+1}} = F^m(n^m)(W^{m+1})^T \frac{\partial \hat{F}}{\partial n^{m+1}}$$

4.24

$$S^m = F^m(n^m)\,(W^{m+1})\;S^{m+1} \qquad 4.25$$

Donde se puede ver por que el algoritmo de backpropagation, (propagación hacia atrás), recibe su nombre:

$$S^M \rightarrow S^{M-1} \rightarrow \ldots \rightarrow S^2 \rightarrow {}^1$$

4.26

Para complementar el algoritmo de propagación hacia atrás, se requiere realizar un último paso, el cual consiste en calcular el punto inicial de S^M. De la forma recurrente de 4.25, se obtiene la relación para el último nivel.

$$S_i^M = \frac{\partial \hat{F}}{\partial n_I^M} = \frac{\partial (t-a)^T (t-a)}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - a_j)}{\partial n_i^N} = -2(t_i - a_j)\frac{\partial a_j}{\partial n_i^M}$$

4.27

Como:

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_j^M)}{\partial n_i^M} = {}^M f(n_j^M)$$

4.28

Entonces

$$S_i^M = -2(t_i - a_i)\overset{M}{f}(n_j^M)$$

4.29

$$S^M = -2\overset{M}{F}((n^M)(t-1))$$

4.30

El algoritmo de propagación hacia atrás, es un algoritmo interactivo diseñado para minimizar el error medio cuadrático entre la salida actual de un perceptrón multinivel y la salida deseada. Para la utilización de este algoritmo se requiere que la función de transferencia sea diferenciable, generalmente para este algoritmo se utiliza la función Sigmoid la cual se define como:

$$F(\alpha) = \frac{1}{1+e^{-(\alpha-\theta)}}$$

4.31

El algoritmo de propagación hacia atrás queda entonces definido como:

Paso 1. Inicialización de los pesos y de los offset.
Se inicializan los pesos y los offset a valores aleatorios pequeños.

Paso 2. Presentar las entradas y salidas deseadas.
Presentar continuamente el vector de entradas X_0 y el vector de salidas deseadas d_0. Si la red es utilizada como clasificador, todas las salidas son cero excepto aquella que corresponde con la clase de la entrada. Las entradas y salidas son presentadas continuamente hasta que la actualización de los pesos se estabilicen.

Paso 3. Calcular la salida actual.
Para calcular la salida actual se utiliza la función Sigmoid como se mencionó anteriormente y el vector y_0 de salida, se calcula con la siguiente relación:

$$Y_0 = f(\sum_{k=0}^{N-1} W_{ki} X_k - \theta)$$

4.32

Paso 4. Actualización de los pesos.
Usando el algoritmo recursivo empezando en los nodos de salida y trabajando hacia atrás hacia el primer nivel, se ajustan los pesos con la siguiente relación.

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta x_i$$

4.33

En esta ecuación W_{ij} de nodo oculto o de la entrada j en el tiempo t, x_i es la salida del nodo i o es una entrada, n es el término de ganancia y delta es el término de error para el nodo j. Si el nodo j es un nodo de salida entonces.

$$\delta_i = y_i(1-y_i)(d_i - y_i) \qquad 4.34$$

Donde d_i es la salida deseada del nodo j e y_i es la salida actual.
Si el nodo j es un nodo interno entonces

$$\delta_i = x_j(1-x_j)\sum_k \delta_k W_{jk} \qquad 4.35$$

Donde el umbral de los nodos ocultos es actualizado de manera similar. La convergencia del algoritmo es más rápida si se agrega un término llamado momento lo cual da como resultado la siguiente ecuación para la actualización de los pesos y se define como:

$$W_{ij}(t+1) = W_{ij}(t) + \eta\delta x_i + \alpha(W_{ij}(t) - W_{ij}(t-1))$$
$$donde \qquad 0 < \alpha < 1 \qquad 4.36$$

Paso 5. Se regresa al paso 2 hasta alcanzar la convergencia de los pesos.

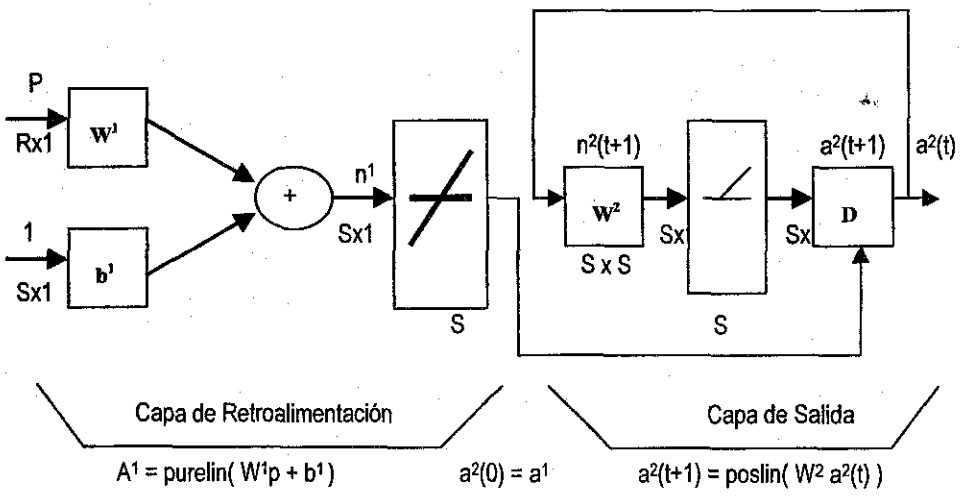


Figura 4.13 Red Hamming

4.13 Red Hamming

La red Hamming puede ser utilizada para el reconocimiento de patrones, ésta utiliza la regla de aprendizaje para clasificar los patrones, su esquema se muestra en la Figura 4.13.
La red de Hamming consiste de 2 niveles. El primer nivel, realiza una correlación entre el vector de entrada y un vector prototipo. El segundo nivel realiza una competencia para determinar cual de los vectores está más cercano al vector de entrada. Para que la red pueda clasificar varios patrones es necesario que se tengan varias neuronas de entrada [15].

Cuando se realiza el reconocimiento de patrones con la siguiente secuencia

$$[p_1, p_2, ..., p_Q] \tag{4.37}$$

la matriz de pesos queda definida como:

$$W^1 = \begin{bmatrix} {}_1W^T \\ {}_2W^T \\ . \\ . \\ . \\ {}_sW^T \end{bmatrix} = \begin{bmatrix} P^T{}_1 \\ P^T{}_2 \\ . \\ . \\ . \\ P^T{}_Q \end{bmatrix}, \qquad b^1 = \begin{bmatrix} R \\ R \\ . \\ . \\ . \\ R \end{bmatrix} \tag{4.38}$$

Cada renglón de W^1 representa un vector prototipo y cada elemento de b^1 es igual al número de elementos de cada vector de entrada, el número de neuronas S es igual al número de vectores prototipo.

La salida del primer nivel es:

$$a^1 = W^1 p + b^1 = \begin{bmatrix} P^T{}_1 P + R \\ P^T{}_2 P + R \\ . \\ . \\ . \\ P^T{}_Q P + R \end{bmatrix} \tag{4.39}$$

Se puede ver que la salida del primer nivel es el producto interno entre el vector de entrada y el vector prototipo más *R*. Donde el producto interno indica que tan semejantes son el vector prototipo y el vector de entrada.

El nivel 2 es un nivel competitivo, las neuronas son inicializadas con las salidas del primer nivel, el cual indica el grado de correlación entre los vectores de entrada y el prototipo. Luego las neuronas compiten entre ellas para determinar un ganador. Después de la competencia, sólo una neurona tendrá su salida diferente de cero. La neurona ganadora indica cual categoría de entradas fue presentada a la red (cada vector prototipo representa una categoría).

4.14 Red Neuronal ADALINE

La red ADALINE mostrada en la Figura 4.14, tiene una estructura semejante al perceptrón, la única diferencia es que la función de transferencia es lineal.

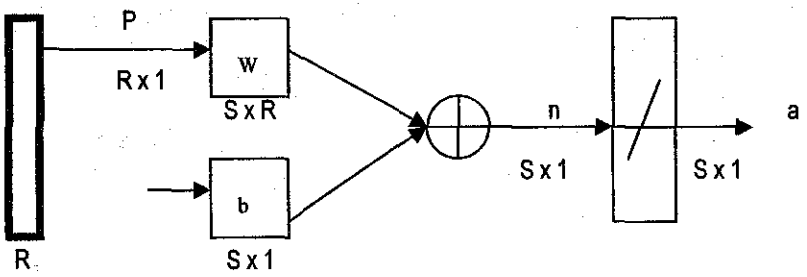


Figura 4.14 Red ADALINE

La salida de la red esta dada por:

$$a = purelin(Wp + b) = Wp + b \qquad 4.40$$

EL ADALINE (o el ALC) es adaptativo en el sentido de que existe un procedimiento bien definido para modificar los pesos con objeto de hacer posible que el dispositivo proporcione el valor de salida correcto para la entrada dada. El significado de correcto a efectos del valor de salida depende de la función de tratamiento de señales que esté siendo elevada a cabo por el dispositivo. El ADALINE es lineal porque la salida es una función lineal sencilla de los valores de entrada.

El ADALINE se puede utilizar para generar una salida analógica utilizando un conmutador sigmoidal, en lugar de binario; en tal caso, la salida se obtendrá aplicando una función del tipo sigmoidal, como la tangente hiperbólica *(tanh (s))* o la exponencial $(1 / 1 + e^{-s})$.

4.14.1 Aprendizaje ADALINE

El ADALINE utiliza aprendizajes OFF LINE con supervisión denominado LMS (Last Mean Squared) o regla del mínimo error cuadrado medio. También se conoce como regla delta porque trata de minimizar una delta o diferencia entre el valor observado y el deseado en la salida de la red; como ocurre con el Perceptrón, sólo que ahora la salida considerada es el valor previo a la

aplicación de la función de activación de la neurona o, si se prefiere, la salida obtenida al aplicar una función de activación lineal.

4.15 Regla de Aprendizaje LMS

La regla de aprendizaje de mínimos cuadrados (Least Mean Square) es un método para hallar el vector de pesos W deseado, el cual deberá ser único y asocia con éxito cada vector del conjunto de vectores o patrones de entrada $\{X^1, X^2, X^3, \ldots, X^L\}$ con su correspondiente valor de salida correcto (o deseado) d_k, $k = 1,\ldots, L$. El problema de hallar un conjunto de pesos W que para un único vector de entrada X da lugar a un valor de salida correcto resulta sencillo, lo que no ocurre cuando se dispone de un conjunto de vectores de entrada, cada uno con su propio valor de salida asociado. El entrenamiento de la red consiste en adaptar los pesos a medida que se vayan presentando los patrones de entrenamiento y salidas deseadas para cada uno de ellos. Para cada combinación entrada salida se realiza un proceso automático de pequeños ajustes en los valores de los pesos hasta que se obtienen las salidas correctas. La regla de aprendizaje LMS minimiza el error cuadrado medio, definido como [15]

$$\varepsilon^2 = \frac{1}{2L}\sum_{k=1}^{L}\varepsilon_k^2 \qquad 4.41$$

Donde L es el número de vectores de entrada (patrones) que forman el conjunto de entrenamiento, y e_k la diferencia entre la salida deseada y la obtenida cuando se introduce el patrón k-ésimo, que en el caso de la red ADALINE, se expresa $\varepsilon_k = (d_k - S_k)$, siendo S_k la salida ALC; es decir:

$$S_k = X_k W^T = \sum_{j=0}^{N} W_j X_{kj} \qquad 4.42$$

La función de error es una función matemática definida en el espacio de pesos multidimensionales para un conjunto de patrones dados. Es una superficie que tendrá muchos mínimos (globales y locales), y la regla de aprendizaje va a buscar el punto en el espacio de pesos donde se encuentra el mínimo global de esta superficie. Aunque la superficie de error es desconocida, el método de gradiente decreciente consigue obtener información local de dicha superficie a través del gradiente. Con esta información se decide qué dirección tomar para llegar hasta el mínimo global de dicha superficie.

Basándonos en el método del gradiente decreciente, se obtiene una regla (regla delta o regla LMS) para modificar los pesos de tal manera que hallamos un nuevo punto en el espacio de pesos más próximo al punto mínimo. Es decir, las modificaciones en los pesos son proporcionales al gradiente decreciente de la función error.

$$\Delta W_j = -(\partial \varepsilon_k / \partial W_j)$$

Por tanto, se deriva la función error con respecto a los pesos para ver cómo varía el error con el cambio de pesos.

Al aplicar la regla de la cadena para el cálculo de dicha derivada:

$$\Delta W_j = -\propto \frac{\partial e_k^2}{\partial W_i} = -\propto \frac{\partial e_k^2}{\partial S_k}\frac{\partial S_k}{\partial W_i} \qquad 4.43$$

Se calcula la primera derivada:

$$\partial \frac{\partial e_k^2}{\partial S_k} = \frac{\partial\left[\frac{1}{2}(d_k - S_k)\right]}{\partial S_k} = \frac{1}{2}\left[2(d_k - S_k)(-1)\right] = -(d_k - S_k) = -e_k \qquad 4.44$$

y por lo tanto queda:

$$\frac{\partial \varepsilon^2}{\partial S_k} = -\varepsilon_k \qquad 4.45$$

Teniendo en cuenta que S_k es la salida lineal:

$$S_k = \sum_{j=0}^{N} W_j X_j \qquad 4.46$$

calculamos la segunda derivada de la expresión de ΔW_i:

$$\frac{\partial S_k}{\partial W_i} = \frac{\partial\left(\sum_{j=0}^{N} W_j X_j\right)}{\partial W_j} = \frac{\partial\left(W_j X_j\right)}{\partial W_i} = Y_{ki} \qquad 4.47$$

así pues, el valor del gradiente de error producido por un patrón dado *(k)* es:

$$\frac{\partial \varepsilon_k^2}{\partial W_i} = -\varepsilon_k X_{ki} \qquad 4.48$$

Las modificaciones en los pesos son proporcionales al gradiente descendente de la función error:

$$\Delta W_i = -\alpha(-\varepsilon_k - x_k) = \alpha\varepsilon_k x_k = \alpha(d_k - S_k)x_{ki}$$

4.49

$$W_i(t+1) = W_i(t) + \alpha(d_k - S_k)x_{ki}$$

4.50

Siendo α la constante de proporcionalidad o tasa de aprendizaje.

En notación matricial, quedaría:

$$W(t+1) = W(t) + \alpha\varepsilon_k x_k = W(t) + \alpha(d_k - S_k)X_k$$

4.51

Esta expresión representa la modificación de pesos obtenida al aplicar el algoritmo LMS, y es parecida a la obtenida para el Perceptrón. Es el parámetro que determina la estabilidad y la velocidad de convergencia del vector de pesos hacia el valor de error mínimo. Los cambios en dicho vector deben ser relativamente pequeños en cada iteración, sino podría ocurrir que no se encontrase nunca un mínimo, o se encontrase sólo por accidente, en lugar de ser el resultado de una convergencia sostenida hacia él.

La diferencia entre ésta expresión y la del Perceptrón está en el valor de error ε_k, que en el caso del perceptrón se refería a la diferencia entre el valor deseado y la salida binaria y_k, y no a la salida S_k de la red.

La aplicación del proceso iterativo de aprendizaje (algoritmo de aprendizaje de una red ADALINE) consta de los siguientes pasos:

1. Se aplica un vector o patrón de entrada X_k, en las entradas del ADALINE.

2. Se obtiene la salida lineal $S_k = X_k W^t$ y se calcula la diferencia con respecto a la deseada.
$$\varepsilon_k = X_k - S_k$$

3. Se actualizan los pesos.
$$W(t+1) = W(t) + \alpha\, \varepsilon_k X_k$$

4. Se repiten los pasos 1 al 3 con todos los vectores de entrada (L).

5. Sí el error cuadrado medio:

$$\varepsilon_k^2 = \frac{1}{2L}\sum_{k=1}^{L}\varepsilon_k^2$$

es un valor reducido aceptable, termina el proceso de aprendizaje; sino, se repite desde el paso 1 con todos los patrones.

Cuando se utiliza una red ADALINE para resolver un problema concreto, es necesario determinar una serie de aspectos prácticos, como el número de vectores de entrenamiento, la forma de generar la salida deseada para cada vector de entrenamiento, la dimensión óptima del vector de pesos, los valores iniciales de los pesos, así como de ser necesario o no un valor del umbral θ. En general, la solución de estas ecuaciones depende del problema concreto que se pretenda resolver, por lo que no se pueden dar respuestas genéricas concretas.

Respecto al número de componentes del vector de pesos, si el número de entradas está bien definido, entonces habrá un peso por cada entrada, con la opción a añadir o no un peso para la entrada del umbral. Incluir este término puede ayudar a la convergencia de los pesos proporcionando un grado de libertad adicional.

La solución cambia cuando sólo se dispone de una señal de entrada. En estos casos, la aplicación más común es el filtro adaptativo; por ejemplo, eliminar el ruido de la señal de entrada, que se muestrea en varios instantes de tiempo de forma que cada muestra representa un grado de libertad que se utiliza para ajustar la señal de entrada a la salida deseada. La idea consiste en utilizar el menor número de muestras (así obtenemos una convergencia más rápida siempre que se obtengan resultados satisfactorios).

La dimensión del vector de pesos tiene influencia directa en el tiempo necesario de entrenamiento (sobre todo cuando se realiza una simulación por computadora), por lo que generalmente se debe establecer un compromiso entre este aspecto y la aceptabilidad de la solución (normalmente, se mejora el error aumentando el número de pesos).

El valor del parámetro tiene una gran influencia sobre el entrenamiento. Si es demasiado grande, es posible que no se produzca la convergencia, debido a que se darán saltos en torno al mismo sin alcanzarlo. Si es demasiado pequeño, alcanzaremos la convergencia, pero a costa de una etapa de aprendizaje larga.

El momento en el que debemos detener el entrenamiento depende, sobre todo, de los requisitos de salida del sistema: se detiene el entrenamiento cuando el error observado es menor que el valor admisible en la señal de salida de forma sostenida. Se suele tomar el error cuadrático medio como la magnitud que determina el instante en el que el sistema medio ha convergido.

4.16 La Red MADALINE

La red MADALINE (Múltiple ADALINE) es una combinación de módulos ADALINE básicos en una estructura de capas que supera algunas limitaciones de la red ADALINE original. El entrenamiento de estas redes no es el mismo que la red ADALINE. El algoritmo LMS podría aplicarse a la capa de salida, puesto que conocemos el vector de salida deseado para cada una de las tramas de entrada de entrenamiento. Sin embargo, lo que se desconoce es la salida deseada para los nodos de cada una de las capas ocultas. Además, el algoritmo LMS funciona para las salidas lineales (analógicas) del combinador adaptativo y no para los digitales del ADALINE. La forma de aplicar la idea del algoritmo LMS para entrenar una estructura del tipo MADALINE pasa por la sustitución de la función de salida por una función continua derivable (la función de umbral es discontinua en 0 y, por tanto no derivable en ese punto).

Existe otro método, conocido como Regla II de MADALINE (MRII), parecido a un procedimiento de acierto y error con una inteligencia adicional basada en un principio de perturbación. El entrenamiento equivale a hacer una reducción del número de neuronas de salida **incorrectas** para cada una de las tramas de entrenamiento que se den como entrada (salida de la red es una serie de unidades bipolares). Este método se puede aplicar mediante el algoritmo:

1. Se aplica un vector a las entradas de MADALINE y se hace que se propague hasta las unidades de salida.
2. Se cuenta el número de valores incorrectos que hay en la capa de salida, denominándose error a dicho número.
3. Para las unidades de la capa de salida:
   - Se selecciona la primera neurona que no haya sido seleccionada antes, cuya salida lineal este más próxima a cero. Esta es la neurona que puede cambiar su salida binaria con el menor cambio de sus pesos y, según el principio de mínima perturbación, debe tener prioridad en el proceso de aprendizaje.
   - Se cambian los pesos de la neurona seleccionada de tal modo que cambie su salida binaria.
   - Se hace que se propague el vector de entrada hacia adelante, partiendo de las entradas y en dirección a las salidas, una vez más.
   - Se admite el cambio de pesos si ha dado lugar a una reducción de error; en caso contrario, se restauran los pesos originales.
4. Se repite el paso 3 para todas las capas, salvo la de la salida.
5. Para todas las unidades de la capa de salida:
   - Se selecciona el par de neuronas que no hayan sido seleccionadas anteriormente y cuyas salidas lineales estén más próximas a cero.
   - Se aplica una corrección de pesos a ambas neuronas para modificar el valor de su salida.
   - Se hace que se propague hacia delante el vector de entrada, desde las entradas hasta las salidas.
   - Se admite el cambio de pesos si ha dado lugar a una reducción del error; en caso contrario, se restauran los pesos originales.
6. Se repite el paso 5 para todas las capas, salvo la de entrada.

Los pasos 5 y 6 se pueden repetir con grupos de tres, cuatro o mayor número de neuronas hasta obtener resultados satisfactorios. Se considera que las parejas son apropiadas para redes que tengan un máximo de 25 neuronas por capa, aproximadamente.

4.17 Consideraciones sobre el Algoritmo de Aprendizaje.

El algoritmo de *backpropagation* encuentra el valor mínimo de error (local o global) mediante la aplicación de pasos descendentes (gradiente descendente). Cada punto de la superficie de la función de error corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local.

Por tanto, uno de los problemas que presenta este algoritmo de entrenamiento de redes multicapa, es que busca minimizar la función de error, pudiendo caer en un mínimo local o algún punto estacionario, no llegando a encontrar el mínimo global de la función de error. Sin embargo, ha de tenerse en cuenta que no tiene porque alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo establecido.

4.18 Control de la Convergencia.

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o lo cerca que se está del punto mínimo. Con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que esto ocurra.

El hecho de elegir un incremento influye en la velocidad con la que converge el algoritmo. Esta velocidad se controla a través de la constante de proporcionalidad o tasa de aprendizaje. Normalmente, será un número pequeño (del orden de 0.05 a 0.25), para asegurar que la red llegue a asentarse en una solución. Un valor pequeño significa que la red tendrá que hacer un gran número de iteraciones. Si la constante es muy grande, los cambios de pesos son muy grandes, avanzando rápidamente por la superficie de error, con el riesgo de saltar el mínimo y estar oscilando alrededor de él, pero sin poder alcanzarlo[15].

Lo habitual es aumentar el valor a medida que disminuye el error de la red durante la fase de aprendizaje. De este modo aceleraremos la convergencia, aunque sin llegar nunca a valores demasiado grandes, que hicieran que la red oscilase alejándose demasiado del valor mínimo. Otra forma de incrementar la velocidad de convergencia consiste en añadir un término llamado momento el cual consistente en sumar una fracción del anterior cambio cuando se calcula el valor del cambio del peso actual. Este término adicional tiende a mantener los cambios de pesos en la misma dirección.

Un último aspecto es la posibilidad de convergencia hacia alguno de los mínimos locales que puedan existir en la superficie de error del espacio de pesos. En el desarrollo matemático que se ha realizado para llegar al algoritmo de retropropagación, no se asegura en ningún momento que el mínimo que se encuentra sea global. Una vez que la red se asienta en un mínimo, sea local o global, termina el aprendizaje aunque el error siga siendo demasiado alto si se alcanza un mínimo local. En todo caso, si la solución es admisible desde el punto de vista de error, no importa si el término es local o global o si se ha detenido en algún momento previo a alcanzar un verdadero mínimo.

En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas ocultas o en los parámetros de aprendizaje o, simplemente, se vuelve a empezar con un conjunto distinto de pesos originales y se vuelve a resolver el problema [7].

4.19 Dimensión de la Red

No se pueden dar reglas concretas para determinar el número de neuronas o el número de capas de una red para resolver un problema concreto. Lo mismo ocurre al seleccionar el conjunto de vectores de entrenamiento. En estos casos, lo único que se puede dar son unas cuantas ideas generales deducidas de la experiencia de numerosos autores.

Respecto al número de capas de la red, en general tres capas son suficientes (entrada-oculta-salida). Sin embargo, hay veces que un problema es más fácil de resolver (la red aprende más deprisa) con más de una capa oculta. El tamaño de las capas, tanto de entrada como de salida, suele venir determinado por la naturaleza de la aplicación. En cambio, decidir cuántas neuronas debe tener la capa oculta no suele ser tan evidente.

El número de neuronas ocultas interviene en la eficiencia de aprendizaje y de generación de la red. No hay ninguna regla que indique el número óptimo, en cada problema se debe ensayar con distintos números de neuronas para organizar la representación interna y escoger el mejor. La idea más utilizada, sobre todo en los sistemas simulados, consiste en tener el menor número posible de neuronas en la capa oculta, porque cada una de ellas supone una mayor carga de procesamiento en el caso de una simulación. En un sistema implementado en hardware, este problema no es crucial, sin embargo, se tendrá el problema de comunicación entre los distintos elementos del proceso.

Es posible eliminar neuronas ocultas si la red converge sin problemas, determinando el número final en función del rendimiento global del sistema. Si la red no converge, es posible que sea necesario aumentar este número. Por otro lado, examinando los valores de los pesos de las neuronas ocultas periódicamente en la fase de aprendizaje, se pueden detectar aquellas cuyos pesos cambian muy poco respecto a sus valores iniciales, y reducir por tanto el número de neuronas que apenas participan en el proceso.

4.20 Inicialización y Cambio de Pesos.

Sería ideal, para una rápida adaptación del sistema, inicializar los pesos con una combinación de valores *(W)* muy cercano al punto de mínimo error buscado. Pero es imposible, porque no se conoce a priori dónde está el punto mínimo. Así se parte de un punto cualesquiera del espacio, inicializando los pesos con valores pequeños aleatorios cualesquiera (por ejemplo 0.5), al igual que los términos umbral, que aparecen en las ecuaciones de entrada neta a cada neurona. Este valor umbral se considera como un peso más que está conectado a una neurona ficticia de salida siempre 1. El término umbral es opcional, pues en caso de utilizarse, es tratado exactamente igual que un peso más y participa como tal en el proceso de aprendizaje.

La expresión de la entrada neta a cada neurona se podrá escribir de la siguiente forma:

$$\mathbf{net} = \sum_{j=1}^{L} w_{kj} x_{pj} + \theta_k \qquad 4.52$$

La modificación de los pesos puede realizarse cada vez que un patrón sea modificado, o bien después de haber acumulado los cambios de los pesos en un conjunto de iteraciones. El momento adecuado para cambiar los pesos depende de cada problema concreto.

Capítulo 5

Desarrollo y Resultados

Introducción

En este capítulo se describe el desarrollo y la utilización de los algoritmos de reconocimiento de voz y del parlante explicados anteriormente, además se define la arquitectura de la red neuronal de tipo LVQ y las características de los vectores de patrones empleados en la etapa de aprendizaje y reconocimiento de la red. En esta parte se desarrollan programas en MATLAB y en Lenguaje C para integrar el sistema de reconocimiento del parlante también se realiza la comprobación de los algoritmos de agrupamiento de datos tales como LBG y K-Medias los cuales sirven como referencia para probar que los programas desarrollados efectivamente realizan el reconocimiento de voz y del parlante. El primer método desarrollado para el reconocimiento del parlante fue el de cuantización vectorial (VQ) para la formación de patrones basados en las características del tracto vocal de cada parlante a través de vectores de LPC mientras que el segundo método desarrollado se basa en la posición del tono contenido en los segmentos de voz de cada parlante; la posición del tono es utilizada para la formación de vectores que posteriormente son utilizados para el aprendizaje de la red neuronal (ANN) y el reconocimiento del parlante.

En la parte final del trabajo se dan los resultados obtenidos en la cuantización vectorial y la red neuronal. Además se dan las conclusiones finales y las posibles mejoras que se pueden realizar a los experimentos, para que el porcentaje de reconocimiento se incremente y a la vez buscar los mejores parámetros de diseño de la red neuronal.

5.1 Desarrollo

Dadas las diferentes aplicaciones de reconocimiento de voz y del parlante el objetivo es desarrollar y verificar algoritmos de reconocimiento de voz y del parlante que puedan ser utilizados como identificador de patrones en aquellas áreas donde se requiera algún tipo de validación del usuario para acceder a recursos restringidos. Para cumplir los objetivos se desarrollaron técnicas diferentes de reconocimiento de voz y a la vez se realizaron pruebas con estos métodos, el primer método desarrollado se basó en los algoritmos clásicos de agrupación de datos como son los métodos LBG y k-medias , el segundo método desarrollado y propuesto en este trabajo para el reconocimiento del parlante fue definido con redes neuronales de tipo LVQ. La formación de los patrones de entrenamiento estuvieron basados en los coeficientes de predicción lineal o coeficientes LPC para el primer método y el segundo método se baso en la extracción del tono de voz por medio de las propiedades de la función de autocorrelación y el algoritmo de recorte central.

Para desarrollar el sistema de reconocimiento del parlante por medio de cuantización vectorial o la simulación de una red neuronal artificial se elaboraron programas en el software de MATLAB y de Lenguaje C, que en su conjunto forman el sistema de reconocimiento.

El reconocimiento del parlante se hace en dos o tres etapas dependiendo si se tiene o no una base de datos de voces. En la etapa uno se graban voces de diferentes parlantes que son utilizadas en el entrenamiento y reconocimiento, la base de datos de voces se constituyó de aproximadamente 1000 palabras diferentes, con 20 parlantes diferentes, algunas grabaciones contienen la misma frase repetida 20 veces y además 5 palabras aisladas. Las frases tienen una duración de 3 a 7 segundos y las palabras aisladas menos de 1 segundo. A la grabación del conjunto de voces se le llamo base de datos UNAM-FI.

La etapa dos es la de entrenamiento donde se extraen las características más representativas de cada uno de los parlantes ya sea por medio de vectores LPC o vectores de tono. La etapa tres es la de reconocimiento que determina que palabra es pronunciada o que parlante es reconocido, por el momento no se considera la etapa de actualización de la bases de datos. Todas las pruebas fueron realizadas en un equipo PENTIUM a 200 Mhz y PENTIUM a 800 Mhz . La frecuencia de muestreo de la señal de voz fue de 11 KHz.

5.2 Sistema de Reconocimiento VQ.

La Figura 5.1 muestra el esquema de formación de los cuantizadores vectoriales para lo cual se utiliza alguna de las técnicas de agrupamiento de datos como el algoritmo LBG o K-medias. Los patrones de referencia o cuantizadores vectoriales serán utilizados para el reconocimiento del parlante.

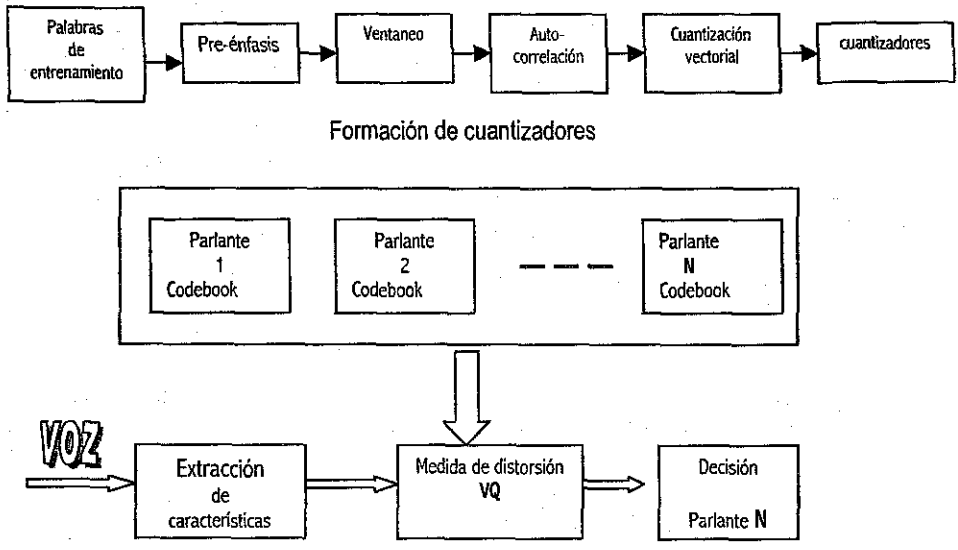


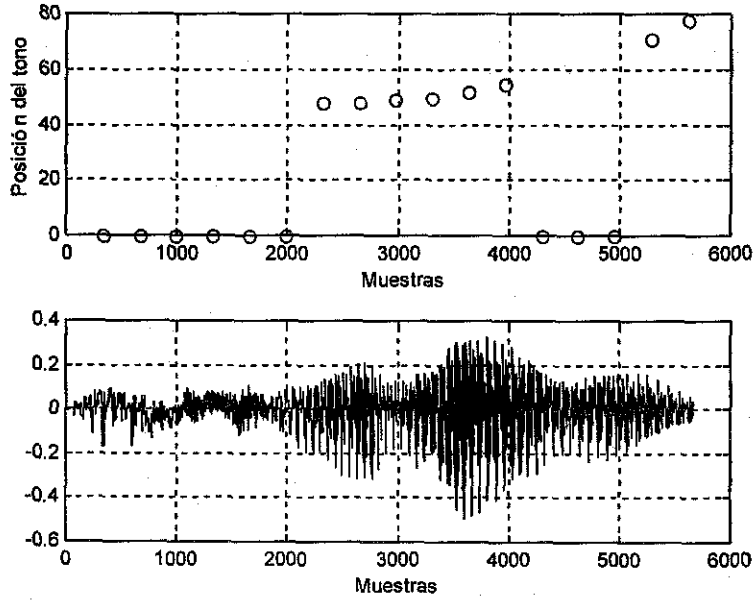
Figura 5.1 Bloques para el reconocimiento del parlante

En la etapa de entrenamiento, en esta etapa de reconocimiento del parlante toma los archivos de voz y se les aplica el procesamiento digital descrito, hasta llegar a formar los vectores de autocorrelación que sirven de entrada para la formación de los cuantizadores vectoriales, el tamaño de los cuantizadores puede variar dependiendo de la duración de las palabras de voz utilizadas, así como del porcentaje de reconocimiento que se requiera. Los tamaños utilizados en este trabajo son de 32, 64 y 128 cuantizadores. Por otra parte el tamaño de los vectores de correlación fueron fijados a 16 elementos.

Una consideración en la formación de los vectores de entrenamiento fue la decisión de tomar todos los vectores de LCP pertenecientes a los sonidos sonoros y no sonoros o solo aquellos que representan sonidos sonoros ya que para el reconocimiento del parlante no es importante considerar todos.

5.3 Extracción de los Vectores de Tono.

Para la extracción del tono del parlante se utilizó el método de recorte central y correlación, uno de los aspectos importantes considerados en esta parte fue el umbral para determinar la posición correcta del tono en el segmento de voz. Para buscar el umbral que proporcionara la mejor aproximación de localización del tono se propusieron 3 umbrales, con un umbral de 0.3 dejaba pasar posiciones que no eran del tono por lo que se fijó un nuevo umbral a 0.4, con éste se logró una mejor aproximación de la ubicación del tono, por último se fijó un umbral de 0.6 teniendo como consecuencia una disminución de las posiciones del tono, finalmente se utilizó para la extracción del tono un umbral de 0.4. La Gráfica 5.1 muestra las posición del tono de voz para un bloque de voz.



Gráfica 5.1. Muestra la posición del tono para un segmento de voz.

5.4 Arquitectura de la Red utilizada en el Reconocimiento.

Una vez establecido el método de extracción del tono se diseñó el tipo de red neuronal a utilizar **basándose en las características de la red de tipo LVQ** cuya arquitectura es la siguiente.

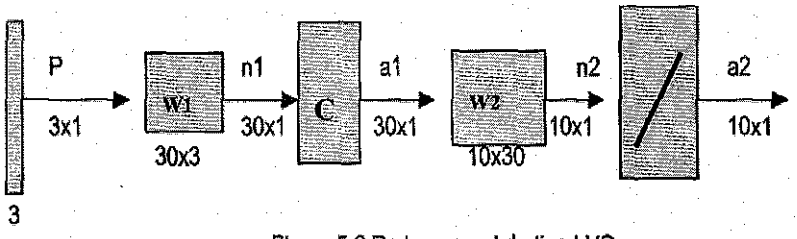


Figura 5.2 Red neuronal de tipo LVQ

En la red de tipo LVQ esta formada por dos niveles; un nivel competitivo y el nivel lineal. En el nivel competitivo **se** recibe como entrada el vector *p* el cual es multiplicado por el vector de pesos *W1* y produce un vector *n1*, los elementos de *n1* son las distancias negativas entre el vector de entrada y la matriz de pesos. El nivel competitivo genera un 0 en todas la neuronas excepto en la neurona ganadora asociada al valor más positivo de *n1*. El nivel lineal trasforma las subclases del nivel competitivo en clases definidas por el usuario. En este tipo de red el primer nivel de neuronas clasifica *n* subclases, mientras que el nivel de salida clasifica *m* clases

5.5 Aprendizaje de la Red

El método de entrenamiento esta basado en el par de entrada/salida

$$\{p_1, t_1\}, \{p_2, t_2\} \ldots \{p_Q, t_Q\}$$

en donde el vector de salida corresponde a la clase deseada.

Para entrenar a la red, un vector de entrada es presentado y la distancia de *P* a cada columna de *W1* es calculada, las neuronas del primer nivel compiten, suponiendo que el *i* elemento de *n* es positivo y la neurona *i* gana la competencia, entonces la función de transferencia produce un 1 como el *i* elemento de a, los demás elementos de *a* son cero. Cuando *a* es multiplicado por los pesos del segundo nivel *W2*, la salida de *a* selecciona la clase de *K* asociada a la entrada, entonces la red ha asignado el vector de entrada *p* a la clase *K* y *a2* es 1.Para ajustar los pesos de *W1* se mueven las columnas al valor de la entrada siempre y cuando la asignación a la clase sea la correcta El algoritmo queda descrito de la siguiente forma:

Si $(a^2_{k^*} = t_{k^*} = 1)$

Se calcula el nuevo valor de *W1* como

$${}_{i^*}\mathbf{IW}^{1,1}(q) = {}_{i^*}\mathbf{IW}^{1,1}(q-1) - \alpha(\mathrm{p}(q) - {}_{i^*}\mathbf{IW}^{1,1}(q-1))$$

Por otra parte si p es asignada a una clase incorrecta tenemos que:

$$(a^2_{k^*} = 1 \neq t_{k^*} = 0)$$

Para calcular el nuevo valor se realiza de la siguiente forma

$$_{i^*}\mathbf{IW}^{1,1}(q) = {}_{i^*}\mathbf{IW}^{1,1}(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{IW}^{1,1}(q-1))$$

La formación de la red para el reconocimiento del parlante queda definida en dos etapas. En la primera etapa de diseño se estableció el tamaño del vector de entrada el cual está formado por la posición de los tonos de la voz de cada uno de los parlantes. Se propusieron vectores de entrada de diferentes tamaños 12, 8, 6, 4,. La segunda etapa de diseño queda formada por la definición del número de neuronas en el nivel de entrada, en ésta etapa se definieron 300 neuronas de entrada con diez neuronas en la etapa de salida, con éste número de entradas se formaron 30 subclases para cada uno de los vectores de tono de cada parlante y 10 clases diferentes. En otra definición de topología se propuso una red con 50 neuronas en el nivel de entrada con 5 subclases y 10 neuronas en la etapa de salida, con un vector de entrada de 6 elementos. La parte difícil en la definición de la red fue acerca de cuántas neuronas deberían existir en el nivel de entrada y el tamaño que deberían tener los vectores de entrenamiento y de reconocimiento.

5.6 Reconocimiento de Palabras Aisladas.

Como primer paso se verificaron los programas realizados para la cuantización vectorial, para este fin se utilizaron como base los archivos de voz que contenían la grabación de los números del 1 al 10, con diez archivos por número para la fase de entrenamiento y diez fueron utilizados para la fase de reconocimiento. En la fase de reconocimiento se utilizaron los algoritmos LBG y K-Medias para la formación de los cuantizadores, con vectores de correlación para el entrenamiento y reconocimiento y como medida de similitud la distancia de Itakura-Saito. El porcentaje de reconocimiento para esta prueba fue del 91.3 %, con lo cual se verificó el funcionamiento correcto de los programas para el reconocimiento de palabras aisladas a través de cuantización vectorial.

5.7 Reconocimiento del Parlante por medio de VQ 32

Para el reconocimiento del parlante se genera un cuantizador vectorial para cada parlante por el método de agrupamiento LBG o K-Medias. En la etapa de reconocimiento un parlante de entrada es reconocido si la distancia total entre un cuantizador y los parámetros de entrada es mínima. Las fases para esta prueba son descritas a continuación.

Fase de entrenamiento:

Para la fase de entrenamiento se tomaron 10 frases diferentes para cada parlante, siendo el número de parlantes para este experimento igual a 7. Posteriormente se aplicó el método de agrupamiento LBG y se obtuvieron 7 code book de tamaño 32; el tiempo de entrenamiento para cada parlante fue aproximadamente de una hora dependiendo del tiempo de duración de cada una de las frases pronunciadas que tenían una duración de 1.85 a 3.25 segundos.

Fase de Prueba:

Una vez obtenidos los code book de cada uno de los parlantes se comenzó la fase de prueba, la cual consistió en un conjunto de 10 frases pronunciadas con duración de 3 a 7 segundos y se aplicó una segmentación lineal, para obtener tres segmentos de igual magnitud, cada uno de estos segmentos se utilizó como frase para la verificación del parlante, los resultados de este experimento se muestran en la Tabla 5.1.

	Parlante1	Parlante2	Parlante3	Parlante4	Parlante5	Parlante6	Parlante7
Parlante1	20	5	5				
Parlante2	2	23	1	4		3	
Parlante3	1	3	16	10			
Parlante4	5	2	4	13	6		
Parlante5					23	7	
Parlante6						24	6
Parlante7			5			4	21

Tabla 5.1 Porcentaje de reconocimiento

La Tabla de porcentaje de reconocimiento muestra los valores de las distancias obtenidas para identificar al parlante, y los resultados obtenidos para cada uno de ellos. Como puede observarse el reconocimiento no fue bueno con un tamaño de code book 32 ya que dicho reconocimiento solo fue del 66.6 %, para 30 segmentos de voz de cada usuario.

5.8 Reconocimiento del Parlante por medio de VQ 64

Se procedió a realizar la siguiente prueba en la que el tamaño de cada code book se incremento de 32 a 64. Además se grabaron frases diferentes para cada usuario de una duración de entre 7 y 11 segundos con la finalidad de tener mayor información de las características de voz de cada parlante.

Para este experimento se realizaron las siguientes pruebas, se generó un code book de tamaño 64 para 10 pronunciaciones de 10 diferentes parlantes, donde cada uno pronunció 10 frases diferentes de duración entre 7 y 11 segundos para el entrenamiento y la clasificación de vectores. El proceso se llevo un día para realizar el cálculo de cada code book. Una vez terminado el entrenamiento se llevaron a cabo las pruebas de reconocimiento, bajo las consideraciones anteriores la parte de verificación presentó demasiados errores, por lo que se hizo una revisión del procedimiento para localizar alguna falla en la programación y buscar alguna mejora. Realizando la revisión en el código, y en la señal de voz se detectó que debido a la duración de las frases utilizadas y del contenido de éstas, los silencios en cada frase influían para realizar una buena verificación ya que formaban parte de los cuantizadores pero no proporcionaban ninguna información de las características de la voz del parlante, para solucionar el inconveniente anterior se

incluyó un procedimiento para eliminar a los segmentos de voz que no pasaran un umbral de energía y con los segmentos resultantes se formaron los cuantizadores vectoriales de los parlantes.

Una vez obtenidos los 10 code book se realizó el reconocimiento, tomando un segundo de las frases pronunciadas por cada parlante formando 30 segmentos de voz por cada uno, los segmentos formados sirvieron para realizar el reconocimiento, los resultados se muestran en la Tabla 5.2

	uno	dos	tres	Cuatro	Cinco	seis	siete	ocho	nueve	diez	Total %
Parlante1	24			4		2					80
Parlante2		30									100
Parlante3		3	22		1			2		2	73.3
Parlante4				29				1			96.6
Parlante5					25		1	1		3	83.33
Parlante6						30					100
Parlante7							29	1			96.6
Parlante8		1						29			96,6
Parlante9									30		100
Parlante10								3		27	90
Total											91.66

Tabla 5.2 Resultados obtenidos con un code book de tamaño 64

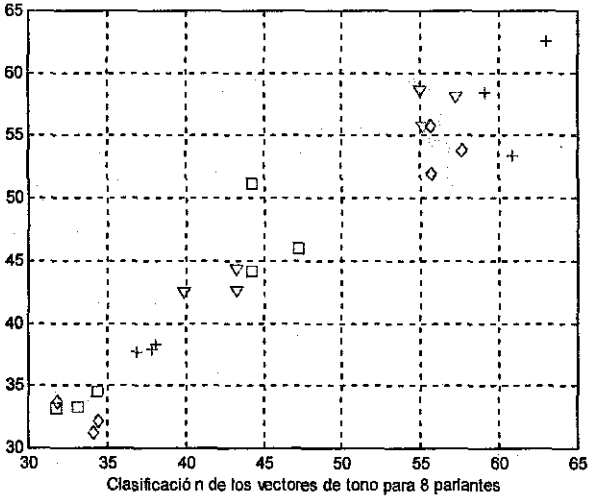
Con las modificaciones realizadas como lo fue la eliminación de los silencios y aumentar el tamaño del code book de cada parlante, el reconocimiento se lleva correctamente en un 91%. Por lo que se concluye en esta parte que el aumento del tamaño de code book trae como consecuencia un aumento en el reconocimiento aunado con la selección de los segmentos sonoros que describen mejor las características vocales de los parlantes.

5.9 Reconocimiento del Parlante a través de ANN de tipo LVQ

En ésta etapa de experimentación se cambian los datos de los patrones de entrenamiento, así como el método de agrupamiento. En lugar de utilizar los vectores de LPC y el método LBG. Se utiliza el método basado en el tono de voz de cada persona que sirve como vectores de entrenamiento. El reconocimiento del parlante independiente del texto se basó en el tono que es una característica de la voz de cada persona. La extracción del tono se realizó de la siguiente manera: de 10 archivos de voz con una duración de 7 a 11 segundos, se utilizaron 5 segundos para la extracción del tono.

Para utilizar el método de autocorrelación en la extracción del tono, se realizó el procesamiento de señal a los 5 segundos de voz y se segmentó en tamaños de 20 ms, a cada uno de los segmentos se les aplicó un umbral de potencia para determinar si el segmento era sonoro o no sonoro con la finalidad de sólo emplear los segmentos que contienen la posición del tono de voz.

A cada segmento sonoro se le extrajo la posición del tono. Con la posición del tono de cada uno de los segmentos se formó un vector que sirve de base en el entrenamiento o reconocimiento de una red neuronal. En el tipo de red neuronal LVQ los pesos del primer nivel *W1* son ajustados con el algoritmo de Kohonen o en su defecto si se conocen las entradas que serán presentadas a la red estas son igualadas a los pesos del primer nivel, por lo tanto como primera prueba, con los vectores de tono obtenidos de tamaño 12 a través del método de autocorrelación se formaron tres vectores que sirven como valores de peso para la red neuronal, en total se formaron 30 vectores de tono para todos los parlantes. Con esto se consigue tener 30 subclases. En la parte competitiva los pesos *W1* de este nivel son inicializados con 30 vectores de tono extraídos de los parlantes entonces se calcula la distancia euclidiana entre las entradas y los pesos del primer nivel. El resultado del cálculo de la distancia es presentado a la entrada a una función competitiva, la cual dará un uno en la neurona cuyo valor sea mayor, es decir el patrón de entrada que más se parezca con algún valor de la matriz de peso *W1*. La matriz de pesos del segundo nivel es inicializada con el número de clases que se requiere a la salida de la red neuronal, por ser una función del tipo lineal la que se encuentra a la salida del segundo nivel, la entrada que active la neurona de salida será la clase a la cual pertenezca. Los patrones de salida formados por la red se muestran en la Gráfica 5.2, donde cada símbolo representa un vector de patrones.



Gráfica 5.2 Distribución de patrones después del aprendizaje

Una vez entrenada la red de tipo LVQ con los promedios de los tonos de cada unos de los parlantes se realizaron pruebas de reconocimiento, para 2 segundos de voz por cada parlante se calculó el tono y se formaron 6 vectores de tamaño 12 que sirvieron como entradas para la red neuronal, para cada parlante se realizaron pruebas de reconocimiento obteniendo los resultados mostrados en la Tabla 5.3.

parlante	1	2	3	4	5	6	7	8	9	10	
1	6										
2	1	1			4						
3			6								
4	4			1							
5					6						
6						3	3				
7			4				2				
8					2			2	2		
9				2					4		
10	3									3	
total											34

Tabla 5.3 Tasa de reconocimiento

Para este tipo de prueba el reconocimiento fue del 56.6 % de reconocimiento, con la red neuronal de tipo LVQ.

Realizando el análisis para encontrar la razón del porcentaje de reconocimiento del 56.6% , se observó que en la topología de red propuesta en la etapa de aprendizaje, sólo un cierto número de neuronas estaban aprendiendo a clasificar los patrones de entrada, mientras que las otras no intervenían en el aprendizaje, por tal razón en el momento de realizar el reconocimiento con los vectores de entrada, no existían patrones suficientes para representar a cada parlante, lo que ocasionaba una tasa baja de reconocimiento. Del grupo de 10 neuronas utilizadas para representar a cada parlante sólo de 2 a 3 de ellas se actualizaban y las restantes nunca formaban parte del aprendizaje, por lo expuesto anteriormente se decidió modificar la topología de la red y el tamaño de los vectores de entrenamiento y reconocimiento. Como primer paso a las modificaciones anteriormente citadas el tamaño del vector paso de 12 elementos a 8 elementos obteniendo los siguientes resultados mostrados en la Tabla 5.4.

	1	2	3	4	5	6	7	8	
1	9								
2		5							
3			10						
4				7					
5					6				
6						5			
7							8		
8								10	
Total									60

Tabla 5.4 Tasa de reconocimiento par vectores de tamaño 8

Para este caso la tasa de reconocimiento fue de 75%, el reconocimiento aumento 18.4 %. Pasando de un vector de 8 a 6 elementos se obtuvieron los valores mostrados en la Tabla 5.5.

	1	2	3	4	5	6	7	8	
1	10								
2		6	2	2					
3		6	2	2					
4		1	2	7					
5					10				
6						10			
7							10		
8								10	
Total									65

Tabla 5.5 Tasa de reconocimiento para vectores de tamaño 6

Para este caso la tasa de reconocimiento fue de 81.25%, el reconocimiento aumentó en un 6.25 %.

Buscando mejorar la tasa de reconocimiento se desarrolló el siguiente algoritmo en el cual se implementan cambios tanto en la topología de la red neuronal, como en el vector de tonos. Los cambios fueron los siguientes :

En la red neuronal se pasó de tener 10 neuronas para cada parlante a sólo 3 por parlante con lo cual el número de neuronas en la capa de entrada disminuyó, así como el número de neuronas en la etapa de salida. Esta disminución resulta benéfica en el tiempo de aprendizaje. Por otra parte se comenzó con un vector de tonos de 6 elementos los cuales se ordenaban y posteriormente se eliminaban el primer y el último elemento, por último las neuronas del nivel de entrada se inicializaban al promedio del conjunto de vectores pertenecientes a cada usuario. A continuación se describe dicho algoritmo.

Algoritmo de entrenamiento de la ANN de tipo LVQ para el reconocimiento

1.- Ordenamiento de los vectores de tono

$$V = [v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \quad v_6]$$

2.- Definir el número de iteraciones y el factor de aprendizaje

N=r LR=0.01

3.- Disminuir el tamaño de los vectores de tono de 6 a 4

$$V = [v_2 \quad v_3 \quad v_4 \quad v_5 \quad]$$

4.- Iniciar la matriz de pesos W1 con el promedio de cada grupo de vectores de cada parlante.

$$W1(k_j) = \frac{1}{10}\sum_{i=1}^{10} x_i \qquad \begin{matrix} j = 1...3 \\ k = 1..10 \end{matrix}$$

5.- Definición de la matriz de pesos **W2**

6.- Presentación de los vectores

$$V_i = [v_2 \quad v_3 \quad v_4 \quad v_5 \quad]$$

7.- Cálculo de la distancia Euclidiana entre el vector V y la matriz de pesos W1

8.- Asignación del vector V a la neurona K

9.- Actualización de los pesos **W1**

Si

$$a_k^2 = t_k = 1$$

$$w1(q) = w1(q-1) + \alpha(v(q) - w1(q-1)$$

En caso contrario

$$a_k^2 = 1 \quad \neq \quad t_k = 0$$

$$w1(q) = w1(q-1) - \alpha(v(q) - w1(q-1)$$

10.- **R=R+1**

11.- Si

R=N termina el aprendizaje

Caso contrario

Regresar a **5**

Una vez programado el algoritmo anterior se procedió a realizar el aprendizaje de la red , el tiempo de aprendizaje no fue mayor a cinco minutos. Terminado el aprendizaje se realizaron pruebas de reconocimiento empezando con 5 parlantes, los datos obtenidos se muestran en la Tabla 5.6.

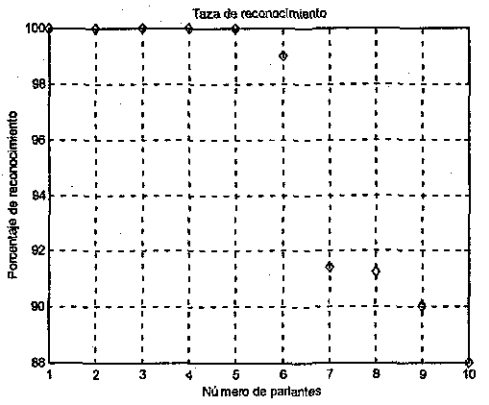
	1	2	3	4	5	6
1	10					
2		10				
3			10			
4				10		
5					10	
Total						50

Tabla 5.6 Reconocimiento para 5 parlantes

Con el algoritmo propuesto se obtuvo un 100% de reconocimiento del parlante con cinco usuarios, para continuar con las pruebas se fue incrementando el número de usuarios hasta llegar a diez obteniendo las tasas de reconocimiento mostradas en la Tabla 5.7 y en la Gráfica 5.3.

Parlantes	Tasa de reconocimiento
5	100%
6	99%
7	91.42
8	91.25%
9	90%
10	90%

5.7 Tasa de reconocimiento para diferente número de parlantes



Gráfica 5.3

Con la tasa de reconocimiento obtenida, se concluye que el algoritmo propuesto funciona para realizar el reconocimiento del parlante.
Existen varios factores que intervienen en el aprendizaje y reconocimiento de la red, uno de los factores es la definición de la matriz de pesos *W1* y el factor de aprendizaje lr . Para verificar lo anterior se definió una matriz de pesos a un porcentaje de los valores de la matriz de entrada. Además se estableció una matriz de entrada P de 24 elementos para formar 4 clases como se muestra en la Tabla 5.8

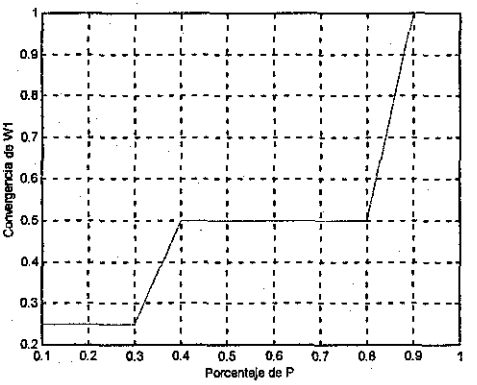
50 50 50 51 51 51 52 52 52	28 28 28 28 28 28 28 28 28	33 33 33 33 33 33 33 33 33	20 20 20 21 21 21 20 20 20

Tabla 5.8 Grupos de tono

El porcentaje que se estableció *W1* respecto a *P* queda definido como:

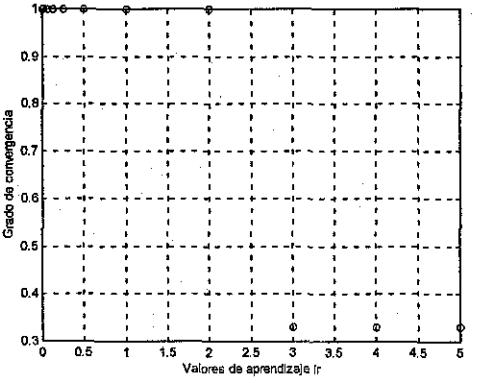
v=[.1 .2 .3 .4 .5 .6 .7 .8 .9 1]

Con estos porcentajes, se realizó el aprendizaje de la red obteniendo la Gráfica 5.4.



Gráfica 5.4 Convergencia de *W1*

La gráfica muestra que ha medida que los valores de inicio de *W1* se inicialicen con un porcentaje de la matriz de entradas, el aprendizaje de la red se aproxima a las salidas deseadas y puede llegar al 100%. También se verificó que a medida que el factor de aprendizaje seleccionado se hacia más pequeño los valores de la matriz de pesos tenían una convergencia a los valores de *P*. La Gráfica 5.5 muestra que a medida de que existe un incremento en el factor de aprendizaje el grado de divergencia es mayor.



Gráfica 5.5 Convergencia de *W1* respecto a *lr*.

Conclusiones

El desarrollo del presente trabajo ha tenido como finalidad la programación de algoritmos de reconocimiento de voz, los cuales pueden utilizarse para el reconocimiento del parlante independiente del texto o dependiente del texto, además los algoritmos pueden ser utilizados para el reconocimiento de palabras aisladas independientes o dependientes del parlante. Con las técnicas utilizadas para el reconocimiento como LBG y K-medias los resultados obtenidos fueron satisfactorios.

Para verificar el método de cuantización vectorial en el reconocimiento del parlante se realizaron varias pruebas, la primera fue de reconocimiento de palabras aisladas en donde se realizó el reconocimiento de diez dígitos pronunciados por el mismo parlante. Este experimento tuvo una eficiencia en el reconocimiento del 91% mostrando que el método empleado, así como la programación de los algoritmos fueron correctos. Basándose en este método se procedió a realizar el reconocimiento del parlante, en este caso se formaron los code book de cada parlante, de tamaño 32. Una vez obtenidos los cuantizadores se procedió a realizar el reconocimiento obteniendo una tasa de reconocimiento por debajo del 30%, ya que el tamaño de los vectores no contenía el número posible de características del tracto vocal de cada parlante por lo tanto se incrementó el tamaño de los code book a 64 code words, para tener más información de cada parlante. La consideración anterior se estableció debido a que la duración de las frases pronunciadas eran de 7 a 11 segundos.

Posteriormente se realizó el entrenamiento para la generación de los code book de tamaño 64 obteniendo una tasa de reconocimiento por debajo del 30%, por lo que se estableció la hipótesis de que los programas desarrollados para el reconocimiento estaban fallando debido a un error de programación. Para localizar el error se hizo una revisión de la programación y del comportamiento de la señal de voz en la cual se pudo detectar que dicha señal contenía silencios muy prolongados y esto ocasionaba que en la formación de los cuantizadores existiera información no útil. Con el método de cuantización vectorial la tasa de reconocimiento aumentó a medida que se eliminaron los vectores que representaban zonas de silencios en la pronunciación de las palabras, ya que por ser palabras de un tiempo de duración de entre 7 y 11 segundos al ser leídas directamente de un texto, existían segmentos de voz que contenían silencios de aproximadamente 200 ms además, las zonas de silencio generaban vectores de voz que no aportaban ninguna información de las características del tracto vocal del parlante. Una vez eliminados los vectores de silencio a través de un umbral, se procedió a realizar una vez más el reconocimiento obteniendo esta vez un 91% de reconocimiento. Con este método se puede concluir que el reconocimiento es bueno considerando que no todos los vectores empleados para la formación de los cuantizadores de cada usuario contienen información valida, además el reconocimiento dependerá de la distorsión de los patrones finales obtenidos por el método LBG o K-medias.

Una vez terminadas las pruebas con cuantización vectorial, se procedió a realizar el reconocimiento del parlante considerando esta vez otra característica de la voz como lo es el tono.

Para la localización del tono se utilizó el método de correlación, aunque se pudo utilizar el método cepstral pero el desarrollo del algoritmo de localización del tono es más complejo. En base al método de correlación se calculó la posición de los tonos de un segmento de voz de diferente duración de 10 pronunciaciones diferentes, con los cuales se formaron vectores por cada uno de los diez parlantes de prueba. Una de las problemáticas de utilizar este método, el cual ocasiona un porcentaje de errores en el reconocimiento del parlante, es que al utilizar frases largas y diferentes en el momento de realizar el aprendizaje de la red, no siempre la posición del tono es la correcta, debido a que dependiendo de la frase pronunciada existe una variación de la posición del tono dependiendo de las palabras que forman la frase y de la rapidez con la que se pronuncian. Esto ocasiona que para un mismo parlante que pronunció frases diferentes, las posiciones del tono no siempre serán semejantes. Como consecuencia, al momento de la formación de sus patrones de referencia existirá una dispersión entre los vectores de un mismo parlante llegando a confundirse con los de otro, por lo que en este caso la tasa de reconocimiento fue baja.

En lo que respecta a la arquitectura de la red dos de los aspectos importantes considerados para la definición de la ANN son: la formación de los vectores de entrenamiento y de reconocimiento, así como el tamaño que estos deberían tener, por lo cual se propusieron diferentes tamaños 12, 8, 6 y 4. Otro aspecto importante en la definición de la arquitectura de la ANN es tomar en cuenta el número de neuronas que deben existir en el nivel de entrada y de salida. La primera propuesta fue de 300 neuronas en el nivel de entrada con lo cual se formaron 30 subclases para cada usuario pero se tuvo que modificar porque en la parte de experimentación no se obtuvo una eficiencia buena por lo que se cambió el número de neuronas de entrada a 50 que forman 5 subclases para cada parlante. El número de neuronas fue modificado en diferentes tamaños hasta llegar a 3 neuronas para cada parlante.

Con los vectores de tono obtenidos se entrenó la red neuronal de tipo LVQ, cuyo objetivo es clasificar patrones de tono. Una vez probado el diseño y la programación de la red, se realizaron pruebas de reconocimiento. El reconocimiento con este método nunca pasó de un 60 % el cual se encuentra muy por debajo de la tasa con respecto al método LBG.

Para mejorar el reconocimiento a través de la red neuronal, se utilizaron para la formación de los vectores de tono, palabras aisladas en lugar de frases en la etapa de entrenamiento y reconocimiento, con lo que se logra que las posiciones del tono tengan mayor correlación entre palabra y palabra, lo anterior ayuda a que disminuya el error en el momento de la formación de los pesos de las neuronas. Con el algoritmo propuesto y desarrollado en la sección 5.6 se logró incrementar considerablemente la tasa de reconocimiento pasando de 56% en las primeras pruebas realizadas con la primera ANN hasta llegar a un 90 % para un grupo de parlantes de 10. Además cabe hacer mención que el algoritmo propuesto cumple con el objetivo planteado de desarrollar un algoritmo de reconocimiento del parlante basado en algún tipo de Red Neuronal Artificial.

En las pruebas realizadas para encontrar la tasa de reconocimiento, se deduce que a medida que el número de parlantes se incrementa, la tasa de reconocimiento disminuye

porque existen vectores (patrones) de un parlante muy cercanos a otro parlante, ocasionando que la clasificación de la red neuronal sea incorrecta por la semejanza entre vectores de diferentes parlantes como es mostrado en la gráfica 5.2, donde se observa que después del aprendizaje los grupos de neuronas pertenecientes a un parlante se pueden cruzar con la de otro parlante.

Otro factor importante que intervino en el reconocimiento es el método utilizado para la obtención del tono de cada parlante, ya que como siempre se obtienen valores enteros de posición del tono, cuando existe demasiada semejanza entre un parlante y otro los valores pueden llegar a ser los mismos por lo cual no existe forma de poderlos distinguir.

Una desventaja del tipo de red propuesta es que al aumentar el número de patrones de entrenamiento siempre es necesario indicar a que neurona de salida corresponden lo que trae como consecuencia que al aumentar o disminuir el número de patrones se modifique el nivel de salida de la red.

Cabe mencionar que el tipo de ANN (LVQ) utilizada en este trabajo es seleccionada por la semejanza que tiene con los métodos tradicionales de agrupamiento de datos como el LBG y K-medias, por esta razón no se realizaron pruebas con algún otro tipo de ANN existente.

Por último se destaca que el mejor método en el trabajo es el de cuantización vectorial ya que con este se logró un reconocimiento del parlante del 91% mientras que con la ANN se logró un reconocimiento del 90% para el mismo número de parlantes. Por lo que es de esperarse continuar con la investigación que busque mejorar el método propuesto de la red neuronal, para que ésta pueda tener un grado de clasificación mayor, y que sea un método donde la tasa de reconocimiento del parlante sea mayor de 90 %. Finalmente el método propuesto de la ANN si puede ser utilizado para el reconocimiento del parlante independiente del texto, así como el método de cuantización vectorial además de que pueden ser aplicados en las áreas donde se requieran.

Bibliografía

[1] Proakis, J and Manolakis, D., Digital Signal Processing, Principles, Algorothms and Applications, Macmillan Publishing, 1992 USA

[2] Rabiner, Lawrence R. And Schafer, R.W, Digital Processing of Speech Signals, Prentice Hall, 1978 USA

[3] Atal, B.S y Hanaver, Suzanne L. Speech Analysis ans Synthesis by Linera Prediction of Speech Wave. Journal Acoustical Society of America. Vol. 50. No. 2. Agosto 1971 p.p 637-655.

[4] Noll, Michael A. Cepstrum Pitch Determination. The Journal of the Acoustical Socciety of America. Vol. 41. No. 2 Agosto 1967. pp. 293-308.

[5] Papamichalis, Panos E. Practical Approaches to Speech Coding. Pretice Hall Inc. Englewood Cliffs, New Jersey. 1987.

[6] Hess, Wolfgang. Pitch determination of Speech Signal, Springer-Verlag. 1983.

[7] Martin T. Hagan y Howard B. Demuth, Neural Networks Design, PWS Publishing . 1996

[8] A Oppenheim and R. Schafer, Discrete Time signal Processing, Englewood Cliffs, NJ, Prentice Hall 1975.

[9] Deller, John, et. Al., Discrete-Time Processing of Speech Signals, Prentice Hall, 1987, USA.

[10] Rabiner, Lawrence, et.al., Fundamentals of Speech Recognitions, Prentice Hall, 1993, USA.

[11] L. Rabiner, A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings IEEE, Vol. 77, No 2, pp. 257-285, Feb 1989.

[12] Y, Linde, A. Buzo and R. Gray, An algorithm for Vector Quantizer Design, IEEE Trans. On Communications, Vol. COM-28, pp 8495, January 1980.

[13] Buzo, H. Martinez, C. Rivera, Discrete Utterance Recognition Based Upon Source Coding Techniques, IEEE Conf. Acoust., Speech Signal Porcessing, Vol.1 pp. 539-542, May 1982.

[14] Redings in Speech Recognition, Alex Waibel, 1990

[15] Simon Haykin, Neuronal Networks, Macmillan, 1994