RECONOCIMIENTO DE LA VOZ MEDIANTE UNA RED NEURONAL DE KOHONEN

Merlo, G; Fernández, V.; Caram, F.; Priegue, R. y García Martínez, R.

Laboratorio de Sistemas Inteligentes
Departamento de Computación. Facultad de Ingeniería. Universidad de Buenos Aires
Paseo Colón 850. 4to Piso. (1063) Buenos Aires. Argentina
{gmerlo, vfernand, fcaram, rpriegue, rgm}@mara.fi.uba.ar

Abstract

El reconocimiento de la voz mediante diversas técnicas tales como cadenas ocultas de Markov y Redes Neuronales es tema de investigación constante, obteniendo resultados de distinta performance según el método elegido.

En el presente trabajo se exhiben los resultados de una experiencia en reconocimiento de voz de un individuo, tomando como patrones a ser reconocidos las cifras decimales (0-9), y utilizando como método una red neuronal de Kohonen.
Luego de una fase de entrenamiento y sintonización, produce con solo cien neuronas un aceptable resultado de reconocimiento (65%).

1. Introducción

El reconocimiento de la voz tiene como dificultad principal reconocer el habla sorteando la gran disparidad entre los registros vocales de distintas personas, según su sexo, edad y pronunciaciones típicas correspondientes a distintas zonas geográficas. Todas estas razones hacen que dos personas que pronuncian la misma palabra, posean patrones frecuenciales y temporales en dicha pronunciación radicalmente distintos, y que en apariencia, no guardan similitud alguna.

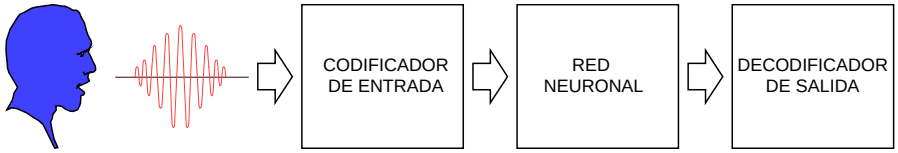
De los múltiples métodos que pudieran ser usados para el reconocimiento de la voz [3][4][6][11], como análisis espectrales [5][8], estadísticos [12], etc., en el presente trabajo se optó por implementar una solución mediante una red neuronal de Kohonen [1][10]. La elección de esta red se debe a que su algoritmo de entrenamiento es bastante simple, lo que determina tiempos de aprendizaje aceptables, en comparación con otro tipo de redes, para una misma cantidad de neuronas. Por otra parte, su característica de mapa auto-organizable aporta una vía natural para asociar neuronas cercanas a pronunciaciones similares dentro del espacio de muestras. El espacio de muestras que debe ser aprendido por la red es el constituido por las diez cifras decimales, y se realizo la experiencia con registros sonoros de tres individuos de sexo masculino adultos.

Para no tener tiempos de entrenamiento ni de reconocimiento muy extensos, se ha optado por implementar la red con solo 100 neuronas. El sistema ha sido implementado y probado en una plataforma PC, y con una placa de sonido Sound Blaster.

2. Estructura básica del sistema

Básicamente el proceso de entrenamiento de la red y reconocimiento de la voz consta de tres partes bien diferenciadas:

A) Grabación de los archivos sonoros y transformación de los mismos en un conjunto de datos o patrones que sean entendibles por parte de la red. Esta tarea de mapeo de datos es realizada por un bloque *codificador de entrada*.
B) Generación y entrenamiento de la red con los patrones obtenidos en la fase anterior (A). Este proceso constituirá la *red neuronal* propiamente dicha.
C) Prueba de la red neuronal obtenida luego del aprendizaje, mediante el reconocimiento de nuevas ocurrencias de los archivos de voz. En todos los casos se utiliza un bloque *decodificador de salida* para transformar el resultado de reconocimiento dado por la red a un elemento de salida entendible, y que será dado por el sistema.



Para la grabación de archivos sonoros se han utilizado formatos estándar VOC y WAV, en sistema MONO, con una resolución de 8 bits y frecuencia de muestreo variable, de la cual nos independiza el módulo codificador de entrada.

2.1. Codificador de entrada

Con este módulo se obtiene una forma de representar los distintos archivos sonoros mediante un patrón que pueda ser manejado por la red neuronal.

Primeramente debemos convenir en que tanto las neuronas como los patrones que deben ser aprendidos por la red neuronal serán representados por vectores fila, de la misma dimensión, y que adoptaran valores reales continuos. Según algunos trabajos realizados, la dimensión adoptada para las neuronas en redes de reconocimiento de voz va de 15 a 100 [1]. En este trabajo hemos adoptado como solución de compromiso entre performance y velocidad, y luego de sucesivas pruebas, una dimensión igual a 20.

Un factor importante a tener en cuenta en el momento del mapeo es poder diferenciar y recortar silencios o zumbidos de fondo previos y posteriores a la palabra pronunciada. El problema de discriminación del habla con respecto al ruido de fondo no es trivial, salvo en casos de gran relación señal a ruido (grabaciones de alta fidelidad en ambientes acústicos, etc.). En tales casos, aun la energía de los sonidos mas bajos o "sordos" (como las fricativas débiles) superan ampliamente la energía del ruido de fondo y la discriminación se puede lograr simplemente por la medición de la energía. Sin embargo en la mayoría de los casos corrientes y de aplicación practica, las condiciones de ruido no son tan favorables, y otro método de discriminación se hace necesario. Un método tal existe y es prácticamente aplicable. Esta forma de discriminación es posible de realizar cuando el archivo de sonido ha sido grabado en un ambiente no extremadamente ruidoso (aunque admite cierto ruido de fondo). Cuando esta ultima condición se cumple, se puede aplicar un sencillo algoritmo que logra este objetivo, y que fue propuesto por Rabiner y Sambur [2].

El algoritmo basa sus cómputos en promedios temporales (mediante una ventana de tiempo rectangular) del valor absoluto de la señal y de los cruces por cero (es decir, las veces en que cambia de signo la señal con respecto a su media).

Debe tenerse en cuenta que el promedio de cruces por cero es un estimador de la frecuencia fundamental (aunque algo rudimentario) que basta para el análisis de comienzo y fin de palabra. Este estimador es el que ayudara a determinar el comienzo y fin de la palabra cuando la misma empieza o termina con sonidos de baja energía y alto valor de frecuencia, como es el caso de las fricativas 'f', 's', 'y', 'z', etc. En esos casos, un análisis único de la función magnitud truncaría en forma incorrecta el sonido; lo que se hace es examinar la función de cruces por cero a partir de los instantes donde seria truncado por la función magnitud, hacia el comienzo (en el caso de la determinación del principio de la palabra) y hacia el final (en el caso de determinación del final de la palabra) del vector de sonido. Si se observa un cambio radical en los cruces por cero (aun en la zona de baja

energía) esta será propuesta como nuevo comienzo (o fin) de la palabra.

Con estas consideraciones el modulo codificador de entrada comienza con la declaración de parámetros, siendo los mismos un vector sonoro MONO de 8 bits por muestra (al que llamaremos SONIDO) y la frecuencia de muestreo a la que fue grabado (Fs), devolviendo un patrón para red (vector PATRÓN).

Como las señales obtenidas por el hardware son siempre positivas, al vector sonoro debemos sacarle la componente de continua (restarle la media estadística). Recién cuando fue extraída la componente continua, calculamos el modulo y los cruces por cero. Dado que un cruce por cero ocurre cuando el signo de dos muestras sucesivas es distinto, podemos calcular rápidamente los cruces por cero tomando el signo del vector sonido, posición a posición y efectuando la siguiente operación:

$$Zn = \sum_{m=-\infty}^{\infty} \left| sgn[x(m)] - sgn[x(m-1)] \right| \cdot w(n-m)$$

Este resultado es guardado en el vector CRUCES. Para esta operación se asume que el signo en t=-1 es 1. CRUCES contiene de esta manera 1's donde se produjo un cambio de signo y 0's donde no lo hubo.

Luego se efectúa un promediado con ventanas temporales unitarias adyacentes de 10 mseg que no se solapan (FRAMES). Este promediado equivale a hacer un filtrado con un filtro pasabajos. De esta manera, de cada ventana se toma un solo valor que representa todo ese intervalo promediado, disminuyendo notoriamente la cantidad de valores con los que se trabaja: estos resultados así obtenidos se guardan en un vector de dimensión mucho menor. Se tendrán así dos nuevos vectores de trabajo que pueden ser interpretados como dos funciones del tiempo que representan al archivo grabado: un vector que representa la función MAGNITUD PROMEDIO (MAGNI_PROM), y un vector que representa la TASA PROMEDIO DE CRUCES POR CERO (ZERO_CROSSINGS).

En el algoritmo original de Rabiner y Sambur se utiliza la función energía y no la función magnitud; la propuesta aquí realizada de utilizar la función magnitud se basa en que por ser la energía el cuadrado de la función magnitud marca mas notoriamente la diferencia entre zonas adyacentes de alta y baja energías, pudiéndose producir un corte incorrecto de la palabra.

Al utilizar la función magnitud (o modulo de la señal) el contorno de amplitud es mas suave y con cortes menos abruptos: los "pozos" de la función magnitud son menos profundos que los "pozos" de la función energía.

Se asume a priori en el algoritmo que los 100 mseg. iniciales del vector de sonido no contienen habla, por lo que se computan la media y la desviación estándar de la función magnitud promedio y tasa de cruces por cero durante este intervalo de tiempo, para dar una caracterización estadística al ruido de fondo (MEDIA_MAGNI_RUIDO, STD_MAGNI_RUIDO, MEDIA_ZC_RUIDO y STD_ZC_RUIDO).

A continuación se procede a analizar la forma de la función magnitud promedio para hallar un intervalo en el cual siempre se exceda un valor de umbral muy conservativo ITU (Interval Thres-hold Upper). A tal fin tomo como ITU un determinado PORCENTAJE (seteable) del máximo valor que toma la función magnitud promedio. Pudiera ocurrir que el sonido hablado fuera tan poco energético que quedara enmascarado por el ruido. Para evitar este inconveniente, y considerando que los valores de magnitud del ruido se hallan en un 99% en la banda comprendida entre su media +/- 3 desviaciones estándar, tomamos como nuevo ITU al máximo entre el porcentaje antes especificado y la media del ruido mas tres desviaciones estándar. En este sentido se procede a comenzar con un PORCENTAJE bajo (es decir, tratando de estar lo mas cerca posible de los limites de la palabra) e ir aumentándolo en 10% sucesivamente si no se cumple la condición de que dentro del intervalo siempre se supere el ITU. De esta manera se llega a establecer el ITU, y con el los valores iniciales de los punteros de PRINCIPIO y FIN de la palabra dentro del vector MAGNI_PROM.

Luego comenzando a trabajar hacia atrás en el vector desde el punto donde fue excedido ITU por primera vez, se halla punto donde la función magnitud promedio cae por

primera vez por debajo de un cierto valor de umbral inferior ITL (Interval Threshold Lower, cuyo valor adoptado en el presente trabajo el igual a la media estadística del ruido mas dos desviaciones estándar). Dicho punto es seleccionado tentativamente como el origen del habla en el vector MAGNI_PROM. Un procedimiento totalmente similar es seguido para determinar el punto final (obviamente con los mismos valores de ITU e ITL). Este procedimiento de doble umbral asegura un suave acercamiento a los extremos inicial y final de la parte hablada del vector sonoro.

Una vez parados sobre los índices hallados por el ITL (PRINCIPIO y FIN en el modulo programado), es razonable pensar que los verdaderos limites de la palabra se hallan fuera de este intervalo. A partir de aquí nos moveremos hacia adelate y hacia atrás con los punteros PRINCIPIO y FIN comparando la tasa de cruces por cero con un limite o umbral IZCT (Interval Zero Crossing Threshold) determinado a partir de las estadísticas de la tasa de cruces por cero del ruido de fondo. Este análisis se limita hasta 25 FRAMES o posiciones anteriores a PRINCIPIO (siguientes a FIN). Si la tasa de cruces por cero supera el umbral 3 o mas veces, el punto inicial PRINCIPIO es movido hasta la primera posición en que fue superado el limite IZCT; en caso contrario el punto inicial (PRINCIPIO) queda definido como lo estaba al comienzo. Un procedimiento similar se sigue para determinar el punto final.

Una vez establecidos los puntos extremos de la parte hablada, simplemente se realiza una transferencia de esa parte intermedia del vector de entrada a otro vector de salida, que en muchos casos llega a ser notablemente menor en tamaño (40 % en algunos casos), lo cual ahorra memoria (factor limitante muy importante al momento de procesar señales) y disminuye mucho el tiempo de computo gracias a la disminución o descarte de información innecesaria y ruidosa.

A partir de aquí queda eliminado el ruido de fondo inicial y final del archivo sonoro, y se procede al mapeo del intervalo hablado en un patrón representativo.

Como ya se dijo, este modulo *mapea* un vector de sonido en un patrón de entrenamiento de la red. Evidentemente, para un correcto funcionamiento del sistema, el patrón o vector obtenido debe ser representativo de la frase dicha: es decir que para una misma palabra varias veces pronunciada por distintas personas (con duraciones, frecuencia fundamental -"pitch"- y energía instantánea distintas) los vectores obtenidos deben ser similares. Este proceso implica también independizarse del tiempo de duración de la palabra.

En el presente trabajo se opto por independizarse del tiempo mediante el uso de la Transformada Rápida de Fourier. De esta manera, para obtener un patrón de red, el modulo codificador de entrada toma un vector de habla y realiza sobre el mismo una serie de sencillas operaciones, que son las siguientes:

- Elimina la componente de continua presente en el vector, simplemente restándole la media estadística del mismo.
- Normaliza el vector, dividiéndolo por la máxima amplitud, de manera tal que no influya demasiado el volumen o amplitud del hablante.
- Ajusta la longitud o dimensión del vector de entrada a un valor que sea potencia de dos, para que de esta manera el algoritmo pueda realizar a continuación el calculo de la FFT de un modo mas rápido utilizando un algoritmo FFT - Radix2 (Fast Fourier Transform - Radix 2), disminuyendo el tiempo de cómputos. Para modificar dicha longitud, se completa al final del vector de entrada con ceros, hasta alcanzar en longitud la potencia de dos mas próxima superior.
- Con la longitud ajustada, calcula una 256-point DFT (Discrete Fourier Transform) con el algoritmo de FFT, y le extrae su modulo o valor absoluto.
- Una vez extraído el espectro discreto de Fourier, se divide el mismo en una cantidad de bandas de frecuencias igual a la dimensión de los patrones, y se realiza la sumatoria de componentes por bandas de frecuencias. Cada una de las sumatorias de las bandas conforman así un valor de coordenada correspondiente en el vector patrón.
- Debido a la disparidad en los resultados de las sumatorias de las distintas bandas de

frecuencias, se extrae el logaritmo natural del vector de sumatorias obtenido en el paso anterior. Esto se realiza en virtud de lograr una cierta compresión en los valores altos, y una expansión en los valores reducidos, efecto totalmente análogo al producido por los circuitos compresores en los sistemas de comunicación; de esta manera la gama de valores de coordenadas no es demasiado extenso.

Con estos pasos realizados, se da como resultado un vector patrón, utilizable por la red neuronal.

2.2. Red Neuronal

La red utilizada es una red de Kohonen, como ya se menciono. No obstante para el aprendizaje de la misma se divide el proceso en dos fases [1], a saber: una primera fase de entrenamiento clásico donde se presentan en forma continua los distintos patrones a la red, y se modifican las posiciones de las neuronas en función de estos; una segunda y ultima fase de ajuste fino, también conocida como algoritmo de Cuantizacion del Vector de Aprendizaje [7][9][13] (Learning Vector Quantization - LVQ) en el cual primero se carátula cada neurona con una etiqueta con el mismo numero del patrón mas próximo, y que en teoría debería representar, y a continuación se van eligiendo al azar pares patron-neurona, corrigiéndose la posición de la neurona mediante un sistema de "premio-castigo", que se explicara mas adelante. Ambos pasos se realizan en el modulo 'som' (por "Self-Organizing Map") que se explica a continuación.

Modulo SOM

Este modulo, encargado de la generación y entrenamiento de la red, toma como parámetros dos matrices (una para entrenamiento clásico inicial y otra para la sintonización fina mediante LVQ) y el numero de neuronas que se desean generar en la red.

Como primera medida se genera la red neuronal en forma aleatoria, para lo cual deben tenerse en cuenta ciertos aspectos referentes a la distribución de los patrones en el espacio multi-dimensional. Luego de realizar multiples gráficos de los patrones (realizando gráficos de las matrices de patrones), se observo una cierta tendencia a la concentración de patrones en una determinada zona intermedia, en una concentracion de tipo Gaussiana. Debido a esta característica, si la generación de neuronas se realiza en el mismo intervalo uniforme para todas las coordenadas, puede suceder que para algunas coordenadas las neuronas estén muy dispersas, mientras que para otras el mismo rango de generación resulte en una concentración muy densa de neuronas. Por esta razón la generación de la red se realiza en forma aleatoria, pero teniendo en cuenta las características estadísticas de cada coordenada: para una neurona generada, cada una de sus coordenadas se obtiene como la suma de la media estadística de la misma coordenada en todos los patrones de entrenamiento, mas un numero aleatorio con distribución uniforme en un intervalo de mas o menos tres desviaciones estándar obtenida en forma estadística de la misma coordenada en todos los patrones de entrenamiento (para así abarcar el 99% del rango de los patrones, ya que se asumió una distribución espacial Gaussiana para los mismos).

A continuación se elige un patrón al azar de la matriz de patrones de entrenamiento, y se calcula la distancia Euclideana entre todas las neuronas y este patrón. Con estas distancias se determina la neurona ganadora. Luego se actualiza tanto la neurona ganadora como sus vecindarias en un radio de alcance determinado, según la regla:

$$M_i(t+1) = M_i(t) + Alfa(t) * Beta(t) * [X(t) - M_i(t)]$$

donde Mi(t) y Mi(t+1) es el vector de una neurona en el tiempo t y (t+1) respectivamente; X(t) es el patrón de aprendizaje presentado a la red en el instante t; finalmente alfa(t) es un factor de peso que disminuye con el tiempo y que simboliza la disminución de la corrección aplicada con el transcurso del tiempo y beta(t), el efecto de achicamiento de la vecindad con el tiempo, hasta que esta solo considera a la ganadora. Este ciclo se repite una determinada cantidad de veces.

Comienza aquí la segunda etapa del algoritmo, para la cual primero deben rotularse las neuronas con la clase o numero a la cual representan. Para rotularlas se procede a comparar cada neurona con todos los patrones de sintonización y se calcula la distancia entre la neurona en cuestión y los patrones. Luego se adopta como etiqueta el patrón mas cercano.

Finalmente comienza la etapa de sintonización fina mediante el algoritmo LVQ. Este algoritmo se basa en un principio muy sencillo: el de premiar o castigar de acuerdo a si el resultado fue positivo o adverso, respectivamente. Cabe mencionar que en el proceso de sintonización fina será preponderante el hecho de tomar los patrones de sintonización en forma aleatoria, para evitar toda tendencia de concentración de neuronas a un patrón en particular.

De esta manera, se procede a tomar de a un patrón por vez y al azar, de la matriz de patrones de sintonización y medir las distancias a todas las neuronas de la red, estableciendo la neurona de mínima distancia. En ese momento descartamos las demás neuronas y verificamos si la clase a la cual había sido asignada la neurona es la misma del patrón. En caso afirmativo, acercamos un poco mas aun dicha neurona al patrón. En caso negativo, la alejamos del patrón. Todas estas actualizaciones se realizan con un coeficiente de corrección de por medio, que debe tener un valor inicial bajo (ya que se trata de una sintonización fina) y se lo va disminuyendo linealmente con las iteraciones hasta llegar a anularse. Se pudo comprobar empíricamente que 100.000 iteraciones da un resultado muy aceptable, sin extender demasiado los tiempos de computo.

Finalizado el calculo de la red, se está en condiciones de probar la misma mediante el reconocimiento de un archivo de voz (.VOC o .WAV) nuevo.

2.3. Decodificador de salida

En el caso particular de esta aplicación, este módulo se limito a asociar cada neurona a una cifra del espacio muestral (0-9). De esta manera, el módulo se limita a imprimir en pantalla el número correspondiente a la clase a la cual pertenece la neurona más excitada, cada vez que se procesa un archivo sonoro.

3. Resultados

Para el entrenamiento y afinación de la red se utilizaron veinte pronunciaciones diferentes de cada cifra, dicha por cinco locutores masculinos adultos; por otra parte, también se utilizaron locutores de idénticas características para la prueba de la red.
Se obtuvo así una tasa de aciertos aceptable para la reducida cantidad de neuronas y muestras vocales con las que se entrenó la red. La prueba se realizó con diez registros vocales de cada cifra, considerando solamente los que fueron bien reconocidos.

NUMERO PRONUNCIADO	TASA DE RECONOCIMIENTO
0	9/10
1	5/10
2	6/10
3	7/10
4	6/10
5	6/10
NUMERO PRONUNCIADO	**TASA DE RECONOCIMIENTO**
6	8/10

7	6/10
8	4/10
9	7/10

4. Conclusiones

El trabajo muestra un aceptable comportamiento de la red neuronal de Kohonen frente a la tarea de reconocimiento de un patrón tan estocástico como lo es la voz. Aunque su tasa de acierto dista aún mucho de las obtenidas mediante otros métodos, existen muchas variables en la red neuronal que pueden ser ajustadas para lograr mejores comportamientos de la misma; entre ellas, se pueden considerar:

- Número de neuronas de la red.
- Dimensión de los patrones y de dichas neuronas.
- Cantidad de locutores y de archivos sonoros utilizados en el entrenamiento.
- Método de codificación o mapeo elegido para la aplicación
- Valores de los coeficientes de aprendizaje y sintonización, y forma de decaimiento en el tiempo.

Todos estos factores interrelacionados contribuirían a mejorar la performance de la red.

5. Referencias

[1] Teuvo Kohonen," The Self-Organizing Map ", *Proceedings of the IEEE*, vol. 78, No. 9, September 1990.
[2] L.R. Rabiner and M.R.Sambur," An Algorithm for Determining the Endpoints of Isolated Utterances ", *Bell Syst. Tech. J.*, Vol. 54, No. 2, pp. 297-315, February 1975.
[3] A. Rosenberg, "Automatic Speaker Recognition: A review", *Proc. IEEE*, vol.64, pp. 475-487, Apr. 1976.
[4] F. K. Soong, A. E. Rosenberg, L. R. Rabiner, and B. H. Juang, "A vector quantization approach to speaker recognition", *in Proc. ICASSP*, pp. 387-390, 1985.
[5] F. K. Soong and A. E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition", *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-36, pp. 871-879, June 1988.
[6] A. L. Higgins, L. G. Bahler, and J. E. Porter, "Voice identification using nearest-neighbor distance measure", in *Proc. ICASSP*, 1993, pp. 375-378.
[7] R. Gray, "Vector quantization", *IEEE ASSP Mag.*, pp. 4-29, Jan 1984.
[8] C. Liu, M. Lin, W. Wang, and H. Wang, "Study of line spectral pair frequencies for speaker recognition", in *Proc. ICASSP*, 1990, pp. 277-280.
[9] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization for speech coding", *Proc. IEEE*, vol. 73, pp. 1551-1587, Nov. 1985.
[10] R. Lippmann, "An introduction to computing with neural nets", *IEEE ASSP Mag.*, pp. 4-22, Apr. 1987.
[11] J. Oglesby and J. S. Mason, "Radial basis function networks for speaker recognition", in *Proc. ICASSP*, 1991, pp. 393-396.
[12] M. Savic and S. K. Gupta, "Variable parameter speaker verification system based on hidden Markov Modeling" in *Proc. ICASSP*, 1990, pp. 281-284.
[13] W. Ren-hua, H. Lin-shen, and H. Fujisaki, "A weighted distance measure based on the fine structure of feature space: Application to speaker recognition", in *Proc. ICASSP*, 1990, pp. 273-276.