Comparación entre Filtrado Óptimo Probabilístico y Filtrado No Lineal mediante Redes Neuronales para Limpieza de Habla Continua con Ruido.

Autor: Rufiner Hugo Leonardo[(*)(#)]

[(*)]*UNER, Laboratorio de Cibernética, Ruta 11 Km 10, 3100 Paraná, Entre Ríos, Argentina,*
[(#)] C.O.N.I.C.E.T., Argentina.
lrufiner@ceride.gov.ar

Palabras clave
Reconocimiento del Habla Robusto • Redes neuronales • Filtrado Óptimo • Denoissing

Resumen

El *Filtrado Óptimo Probabilístico* (POF) constituye una técnica de limpieza de ruido en el dominio de las características extraídas de la señal de habla continua con el fin de implementar sistemas de *Reconocimiento Automático del Habla Continua Robustos* (RACSR). El procesamiento generalmente empleado para extraer las características es el de las *Derivadas de los coeficientes Cepstrales en escala de Mel* (Delta MFCC). Es sabido que la relación de mapeo requerida entre el espacio de las características de las señales ruidosas y el de las señales limpias es no lineal. POF constituye un mapeo multidimensional lineal por tramos que requiere tener muestras apareadas (*stereo*) de las señales con y sin ruido. Las *redes neuronales artificiales* (ANN) son aproximadores universales de funciones arbitrarias y por lo tanto se convierten en una alternativa interesante para la solución de este problema. Las ANN permiten el mapeo directo entre ambos espacios de manera no lineal lo que constituiría un *verdadero filtro no lineal.* Sin embargo los procedimientos utilizados para entrenarlas son generalmente lentos y muy propensos a caer en mínimos locales. Otra posibilidad de las redes es incluir algún tipo de dinámica en su estructura para aprovechar las relaciones temporales existentes entre los patrones. Una característica deseable en este tipo de técnicas de adaptación al ambiente es su capacidad de ser entrenados con poca cantidad de datos (con relación al total utilizado para entrenar al sistema completo de reconocimiento). En el presente trabajo se realiza una comparación entre ambos métodos sobre un conjunto de datos tomados del Corpus de habla continua en español LATINO 40, los cuales se mezclaron con ruido blanco obtenido de la base NOISEX-92 en diferentes proporciones. Las señales filtradas mediante ambos métodos se pasaron a través de un reconocedor basado en *Modelos Ocultos de Markov* (HMM) *Continuos de Mezclas Gaussianas* entrenados con habla limpia de la misma base de datos, para confrontar los niveles de reconocimiento de ambas técnicas.

Introducción

En muchas situaciones prácticas un reconocedor automático del habla tiene que operar en varios ambientes acústicos diferentes pero bien definidos. Por ejemplo, la misma tarea de reconocimiento puede llevarse a cabo usando diferentes micrófonos o canales de transmisión . En esta situación puede no resultar práctico recoger un corpus de habla para entrenar nuevamente a los modelos acústicos del reconocedor. Para aliviar este problema, se han propuesto varios algoritmos que mapean características del habla entre dos espacios acústicos o que "limpian" la señal contaminada con ruido. Estas soluciones constituyen alternativas en el lado del espacio de características de la señal o bien directamente en el espacio de la señal [San96]. Otra opción constituye la variación, ajuste o tunning de los parámetros del modelo acústico o del clasificador empleado, lo cual constituye una transformación en el dominio de los parámetros del modelo acústico. En ambos casos los algoritmos son entrenados usando una base de datos preferentemente pequeña grabada en el nuevo ambiente de prueba o aplicación, o bien simultáneamente en ambos ambientes. En la Figura 1 se muestra un esquema con las distintas fuentes de variabilidad de la señal de voz y distintos tipos de soluciones propuestas. En el presente trabajo nos ocuparemos de una comparación entre dos posibles métodos de mapeo en el dominio de las características, un mapeo lineal por tramos basado en un filtro óptimo probabilístico (POF), y un mapeo no lineal con redes neuronales (NNNLF) (ver Figura 2).
Este trabajo se organizará de la siguiente forma: a continuación se presentarán los fundamentos de ambas técnicas (POF y NNNLF), sus aspectos comunes y diferencias. Posteriormente se describirán los datos utilizados y la forma en la que se llevaron a cabo los experimentos. En

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

seguida se expondrán los resultados obtenidos y finalmente se presentará la discusión y las conclusiones que se derivan de este trabajo.

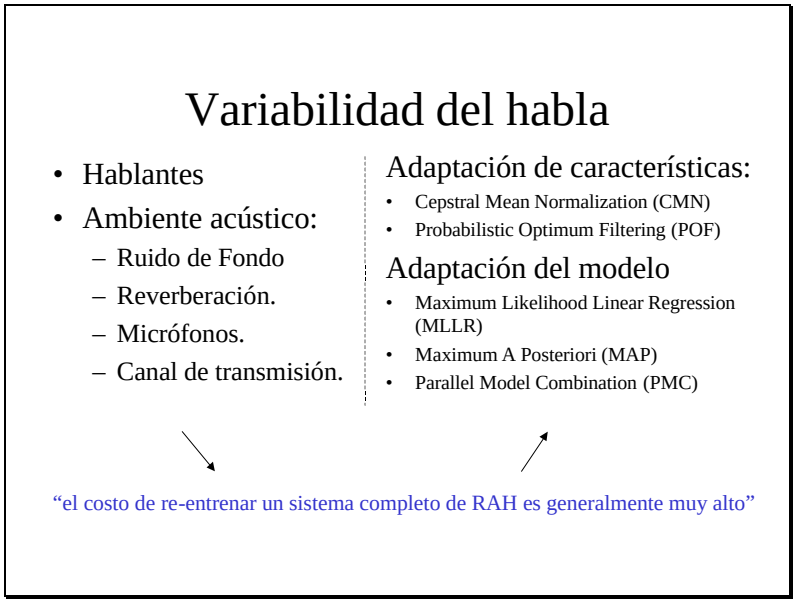


Figura 1: Fuentes de variabilidad del habla y soluciones propuestas

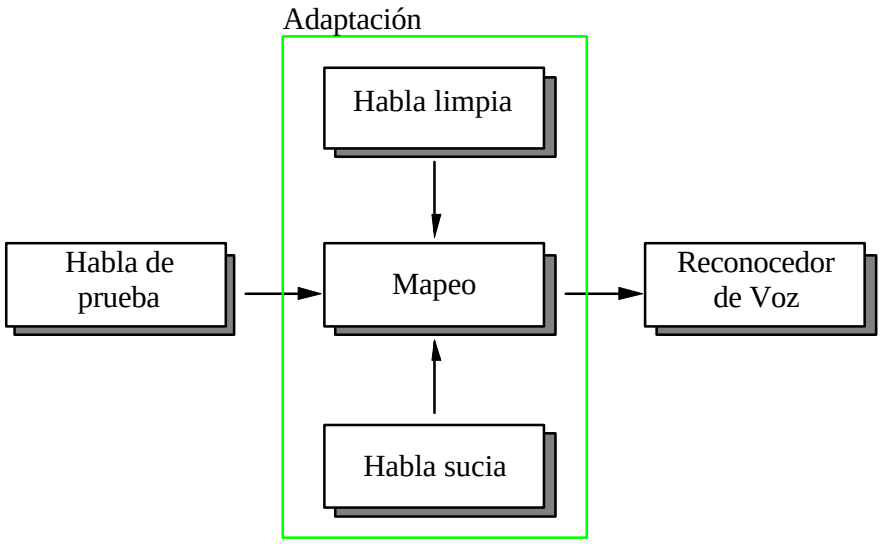


Figura 2: Esquema general del mapeo

Métodos

Filtrado Óptimo Probabilístico

En el caso de ruido aditivo homogéneo estacionario, podemos derivar una estimación de cuadrados mínimos de las características del habla limpia (log-energías del banco de filtros) usando un modelo para la forma en la que las características cambian en la presencia de ruido [ErW89-90]. En estos algoritmos, el espectro del habla estimado es una función de la SNR espectral global, la SNR espectral instantánea, y la envolvente espectral global de la señal del habla. Sin embargo, se sabe [NeW94] que la relación entre los dos espacios de características simultáneas es no lineal. Por consiguiente en [NeW94] se usa un modelo lineal por tramos para relacionar los dos espacios de características. En la literatura ha habido varios algoritmos que se han enfocado en entrenar experimentalmente un mapeo entre las características ruidosas y las limpias ([ErW89], [ErW90], [JuR87], [GCR90], [NGR92], [Ace90])

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

El algoritmo de POF

El algoritmo de mapeo está basado en una transformación probabilística lineal por tramos del espacio acústico llamada Filtrado Óptimo Probabilístico (POF). Vamos a asumir que el reconocedor está entrenado con datos grabados con un micrófono de alta calidad cerca del hablante (habla limpia) y los datos de prueba son adquiridos en un ambiente acústico diferente (habla ruidosa). Nuestra meta es estimar un vector de características limpio $\hat{x}_n$ dado su correspondiente vector de características ruidoso y_n donde n es el índice del frame. Para estimar el vector limpio cuantizamos vectorialmente (VQ) el espacio de las características limpias en I regiones usando el Algoritmo de Lloyd generalizado [LBG80]. A cada región de VQ se le asigna un filtro transversal multidimensional (ver Figura 3). El error entre el vector limpio y los vectores estimados producidos por el *i-ésimo* filtro está dado por:

$$e_{ni} = x_n - \hat{x}_{ni} = x_n - W_i^T Y_n \tag{1}$$

donde el e_{ni} es el error asociado con la región i, W_i es la matriz de coeficientes del filtro, y Y_n es la línea de retardo de los vectores ruidosos. Expandiendo estas matrices tenemos:

$$W_i^T = \left[A_{i,-p} \cdots A_{i,-1}\ A_{i,0}\ A_{i,1} \cdots A_{i,p}\ b_i\right] \tag{2}$$

$$Y_n^T = \left[y_{n-p}^T \cdots y_{n-1}^T\ y_n^T\ y_{n+1}^T \cdots y_{n+p}^T\ 1\right] \tag{3}$$

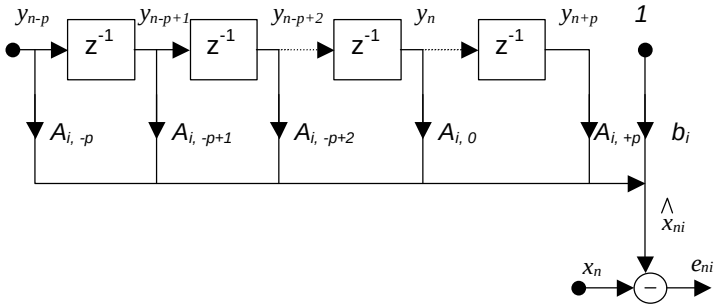


Figura 3: Filtros transversales multi-dimensionales para el cluster i

El error condicional en cada región se define como:

$$E_i = \sum_{n=p}^{N-1-p} \left\|e_{ni}\right\|^2 p(g_i|z_n) \tag{4}$$

donde el $p(g_i|z_n)$ es la probabilidad de que el vector limpio x_i pertenezca a la región g_i dado un vector condicional arbitrario de características ruidosas z_n. Nótese que la característica ruidosa condicionante puede ser cualquier vector acústico generado a partir del frame de habla ruidosa (o directamente este último). Por ejemplo, puede incluir una estimación de la relación señal ruido (SNR), energía, energía cepstral, cepstrum, etc.,

La función densidad de probabilidad condicional $p(z_n|g_i)$ se modela como una mezcla de I distribuciones Gaussianas. Cada distribución Gaussiana modela una región de VQ. Los parámetros de las distribuciones (vectores medios y matrices de covarianza) se estiman usando los vectores z_n correspondientes asociados con esa región. Las probabilidades a posteriori $p(g_i|z_n)$, se calculan usando el teorema de Bayes y los pesos de la mezcla, $p(g_i)$, se estiman usando el número relativo de vectores limpios de entrenamiento que se asignan a una región de VQ dada.

Para calcular los filtros óptimos en el sentido del error cuadrático medio, minimizamos el error condicional en cada región de VQ. El vector del error medio mínimo es obtenido tomando el gradiente de E_i definido en la Ec.(4) con respecto a la matriz de coeficientes del filtro e

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

igualando toda la matriz gradiente a cero. Como resultado, la matriz de coeficientes del filtro óptimo tiene la forma, $W_i = R_i^{-1} r_i$ donde:

$$R_i = \sum_{n=p}^{N-1-p} Y_n Y_n^T \, p(g_i | z_n) \tag{5}$$

$$r_i = \sum_{n=p}^{N-1-p} Y_n x_n^T \, p(g_i | z_n) \tag{6}$$

es una matriz de auto-correlación probabilística no singular, y:
es una matriz de correlación cruzada probabilística.
El algoritmo puede entrenarse completamente sin supervisión y no requiere ninguna información adicional a las formas de onda simultáneas.
La estimación en tiempo de corrida del vector de características limpias puede ser calculada integrando las salidas de todos los filtros como sigue:

$$\hat{x}_n = \sum_{i=0}^{l-1} W_i^T Y_n \, p(g_i | z_n) = \left\{ \sum_{i=0}^{l-1} W_i^T \, p(g_i | z_n) \right\} Y_n \tag{7}$$

Tabla 1: Lista de Símbolos utilizados.

Símbolo	Dimensión	Descripción
n	1	índice del frame
i	1	índice de la región
L	1	tamaño del vector de características
M	1	tamaño del vector de características condicionado
N	1	número de frames de entrenamiento
l	1	número de regiones de VQ
p	1	retraso máximo del filtro
e_{ni}	Lx1	vector del error de estimación
x_n	Lx1	vector de características limpias
$\hat{x}_n$	Lx1	vector de característica limpias estimado
y_n	Lx1	vector de características ruidosas
z_n	Mx1	vector de características ruidosas condicionando
m_i	Mx1	vector medio de la gaussiana i
S_i	MxM	matriz de la covarianza de la gaussiana i
W_i	(2p+1)L+1 x L	matriz de coeficientes de los filtros transversales
Y_n	(2p+ 1)L+1 x 1	linea de retardo del vector de entrada
A_{ik}	LxL	matriz multiplicativa (tap)
b_i	Lx1	matriz aditivo (tap)
R_i	(2p+ 1)L+ 1 x (2p+1)L+1	matriz de auto-correlación
r_i	(2p+l)L+1 x L	matriz de correlación cruzada

Una lista de los símbolos utilizados se muestra en la Tabla 1. Para su implementación se ha dividido el algoritmo POF en dos partes, una que corresponde a la fase de entrenamiento de los parámetros del filtro (cuyos pasos se detallan en la Tabla 2), y otra que corresponde a la fase de corrida sobre datos no vistos durante el entrenamiento para generar las estimaciones de los vectores limpios (cuyos pasos se detallan en la Tabla 3). En la Tabla 4 se describe el algoritmo utilizado para realizar la VQ. Con fines ilustrativos en las Figuras 3-9 se muestran gráficos de las distintas variables del proceso durante el entrenamiento de un filtro sobre la oración af14_001 de Latino40, 10 clusters y orden de los filtros igual a 2. Observesé, por ejemplo, como $P(z_n)$ aumenta especialmente en los períodos de silencio donde existe una menor SNR instantánea (Figura 7).

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

Tabla 2: Algoritmo de entrenamiento POF

1. Leer datos de las características de origen y destino (y, x).
2. Fijar orden del filtro p y cantidad I de clusters a utilizar.
3. Realizar una VQ para cada frame de los datos destino del mapeo:
   a. Entrenar parámetros del cuantizador mediante algoritmo de Lloyd (calculo centroides) (Tabla 4).
   b. Cuantizar vectorialmente los datos (asignar los frames a los distintos clusters generados)
4. Calcular probabilidad a priori de que un vector limpio pertenezca al cluster g_i , $P(g_i)$, como la razón entre el número de frames que caen en cada cluster y el número total.
5. Calcular parámetros de la probabilidad del vector de características ruidosas condicional zn dado que el vector limpio xi pertenece a la región g_i , $P(z_n|g_i)$ (se asume aquí que el vector de característica ruidosas condicional es directamente $z_n=y_n$):
   a. Calcular vectores medios de los vectores sucios y_n para cada cluster.
   b. Calcular matrices de covarianza de los vectores sucios y_n para cada cluster.
6. Calcular la probabilidad de ocurrencia de los vectores sucios $P(z_n)$ o función de verosimilitud como una mezcla Gaussiana utilizando g_i y los parámetros para $P(z_n|g_i)$:
$$P(z_n) = \sum_i P(g_i) \cdot P(z_n | g_i)$$
7. Calcular $P(g_i|z_n)$ utilizando Bayes como: $P(g_i|z_n) = (P(z_n|g_i) \,.\, P(g_i)\,) \,/\, P(z_n)$
8. Calcular matrices de correlación probabilísticas mediante las ecuaciones 5 y 7
9. Calcular coeficientes del filtro como $W_i = R_i^{-1} r_i$
10. Guardar parámetros del filtro entrenado.

Tabla 3: Algoritmo de corrida POF

1. Leer datos de las características de origen y.
2. Cargar parámetros del filtro entrenado previamente: orden del filtro p, cantidad de clusters I, coeficientes del filtro W, medias μ, matrices de covarianza Σ, y probabilidad a priori $P(g_i)$.
3. Calcular la probabilidad de ocurrencia de los vectores sucios $P(z_n)$ o función de verosimilitud como una mezcla Gaussiana utilizando $P(g_i)$ y los parámetros para $P(z_n|g_i)$:
$$P(z_n) = \sum_i P(g_i) \cdot P(z_n | g_i)$$
4. Calcular $P(g_i|z_n)$ utilizando Bayes como: $P(g_i|z_n) = (P(z_n \mid g_i) \,.\, P(g_i)\,) \,/\, P(z_n)$
5. Armar la línea de retardo de los vectores sucios Y mediante la ecuación 3.
6. Calcular el vector de características limpio estimado integrando las salidas de todos los filtros mediante la ecuación 8.
7. Si tengo datos destino calcular medidas de desempeño (como ECM, SNR, etc).

Tabla 4: El algoritmo de Lloyd generalizado.

1. Comenzar con un libro de códigos (codebook) inicial C_1.
2. Repetir
   (a) Dado un codebook (conjunto de agrupamientos (clusters) definido por sus centroides) $C_m = \{y_i ; i = 1, \cdots, N\}$, redistribuir cada patrón del conjunto de entrenamiento en uno de los clusters en C_m, seleccionando aquel cuyo centroide este más cerca de cada patrón.
   (b) Recalcular los centroides para cada uno de los clusters recién creados para obtener el nuevo codebook C_{m+1}.
   (c) Si se generó una celda vacía en el paso anterior, se realiza una asignación alternativa del vector de código (en lugar del cálculo del centroide). Por ejemplo partiendo otro cluster en dos y reasignando la partición a la celda vacía.
   (d) Calcular la distortion promedio para C_{m+1}.

Hasta que la distorsión haya cambiado solo una pequeña cantidad desde la última iteración:
$$\frac{(D_m - D_{m+1})}{D_m} < e, \;\; 0 \le e \le 1$$

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

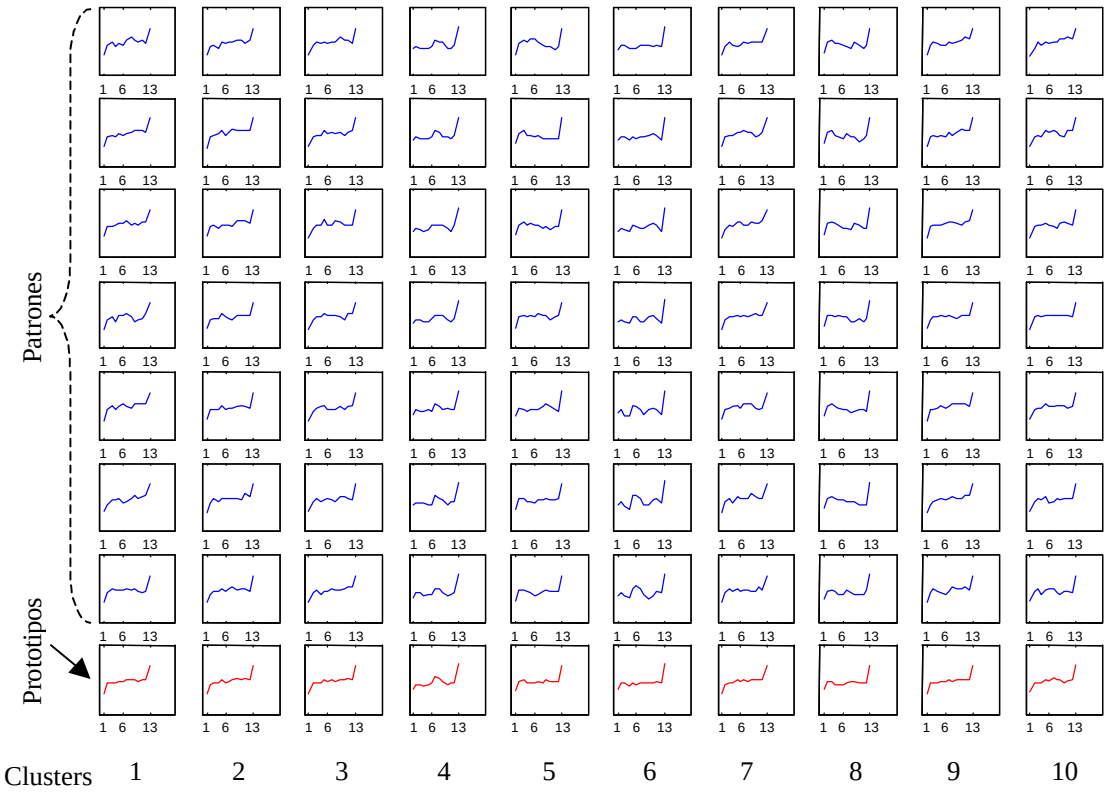


Figura 4: Ejemplos de vectores sucios y Medias de los clusters.

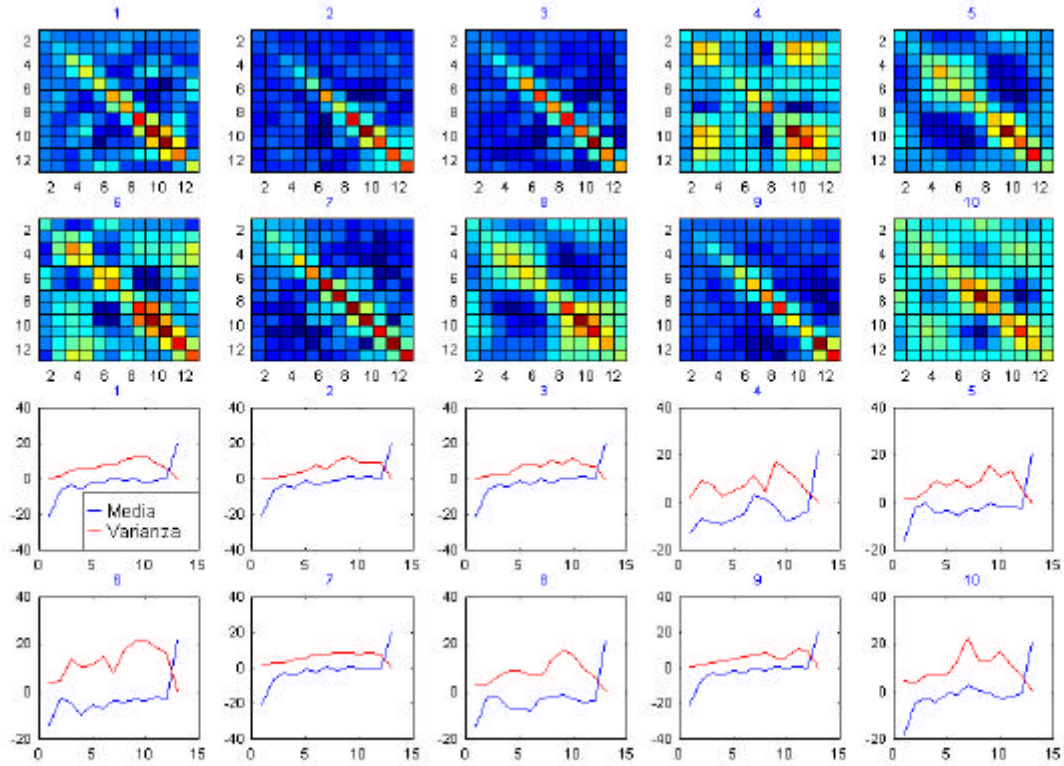


Figura 5: Matrices de covarianza, Medias y varianzas de los clusters.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

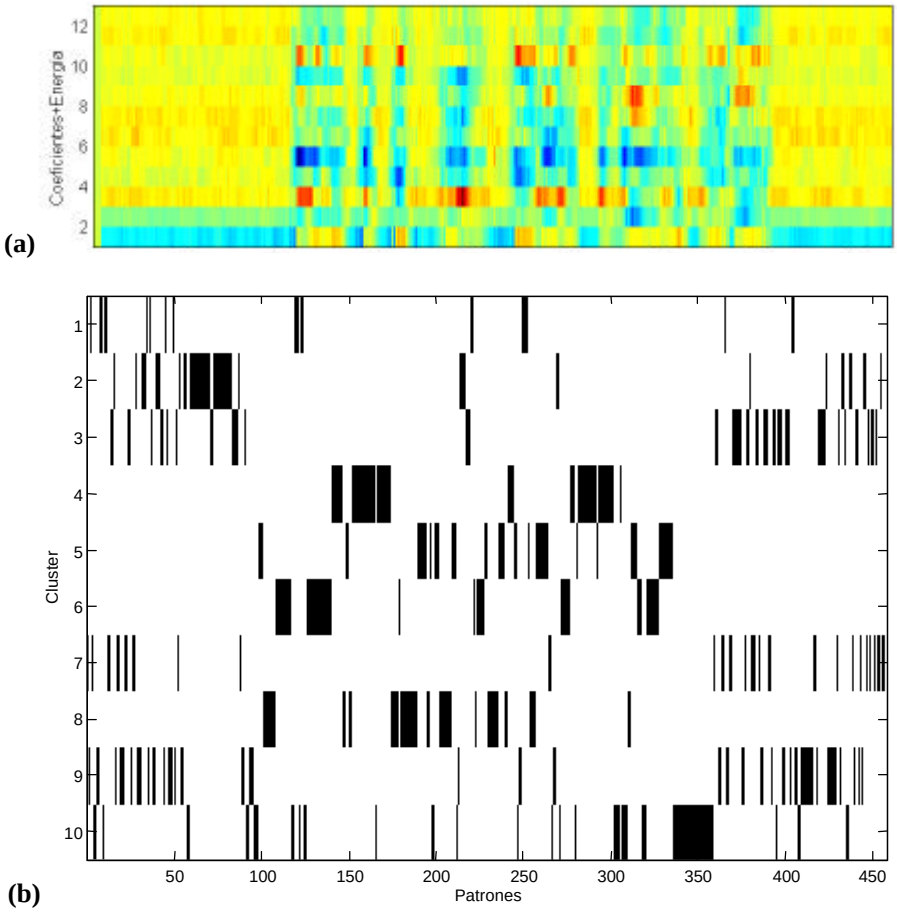


Figura 6: Resultado de la cuantización vectorial: (a) vectores o frames de entrada, (b) asignación de los vectores o patrones a cada cluster.

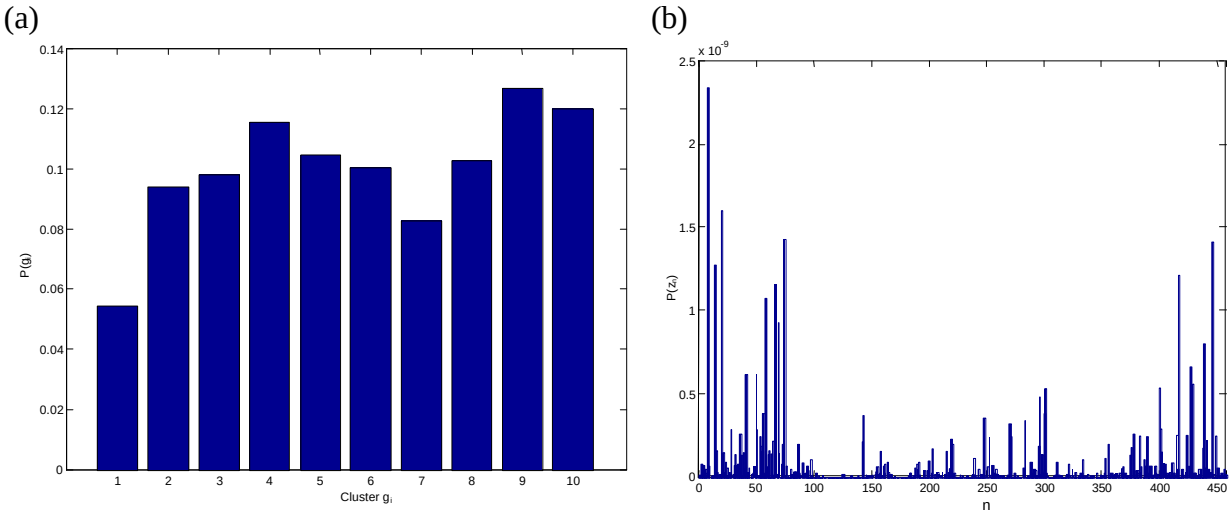
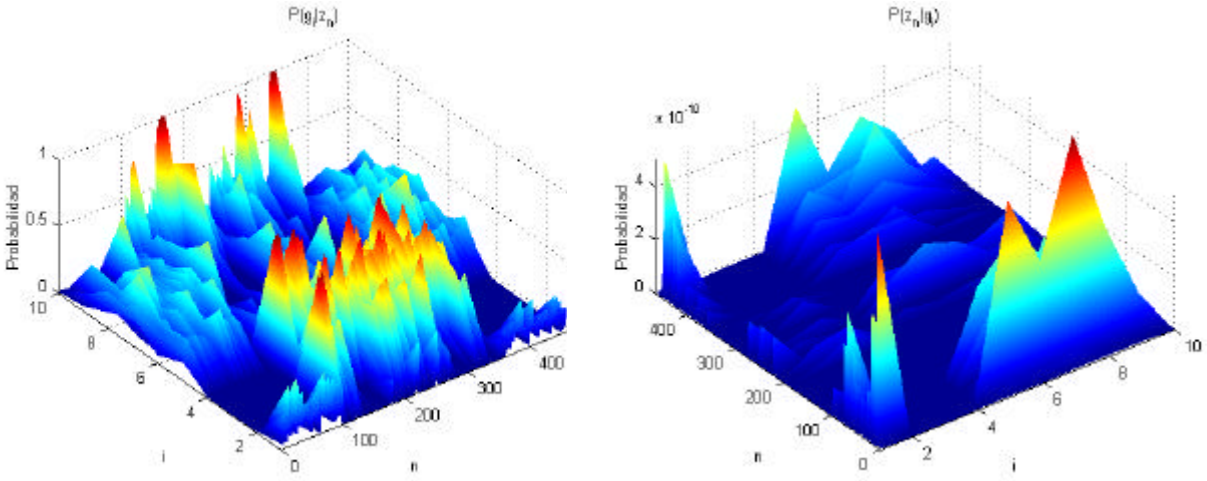


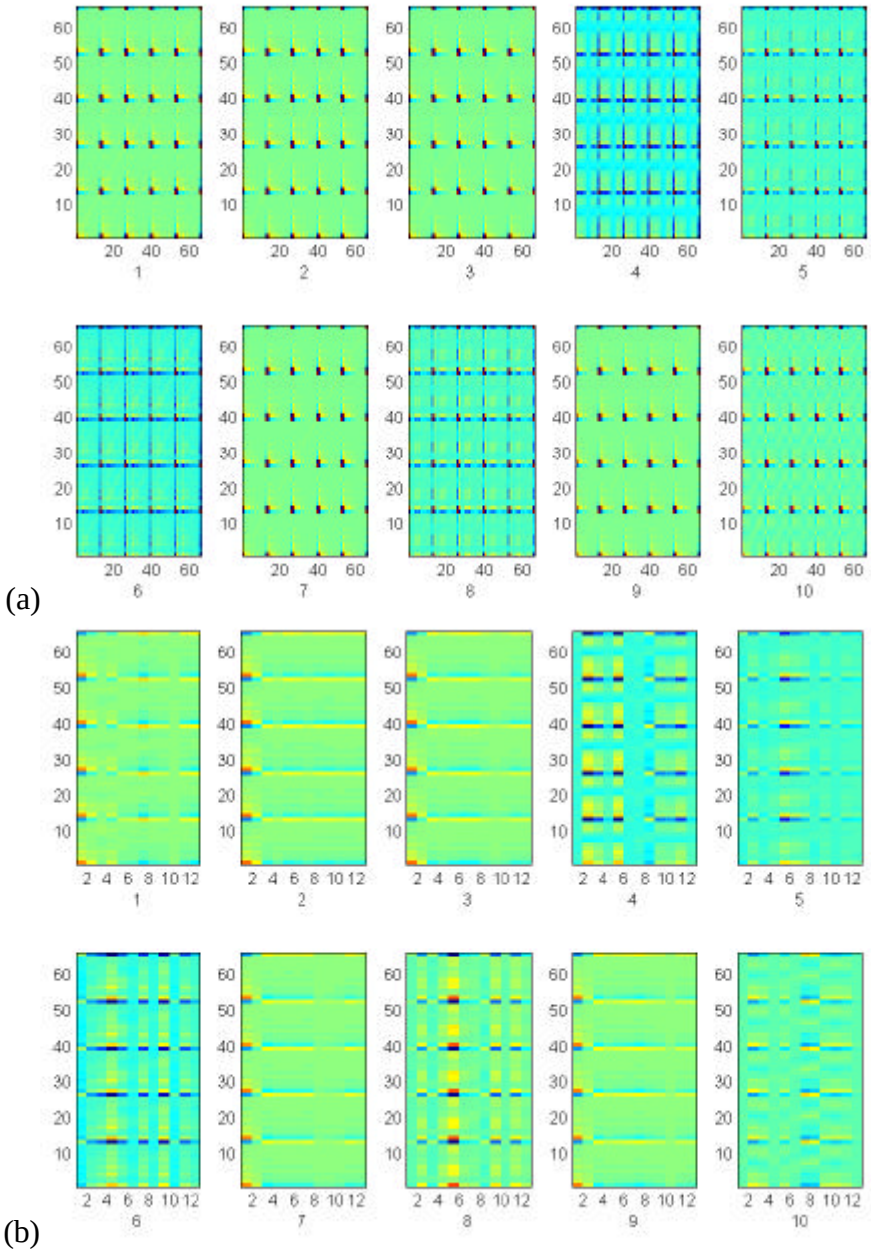
Figura 7: (a) **Probabilidad que un vector pertenezca al cluster g_i** , (b) **Probabilidad de ocurrencia de los vectores sucios $P(z_n)$.**

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.



(a) (b)

Figura 8: (a) Probabilidad condicional de pertenecer al cluster g_i dado el vector de sucio z_n, $P(g_i|\ z_n)$. (b) Probabilidad condicional de los vectores sucios dado que x pertenece al cluster g_i, $P(z_n|g_i)$



(a)

(b)

Figura 9: (a) Matrices de Autocorrelacion R_i, (b) Matrices de Correlacion Cruzada r_i

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

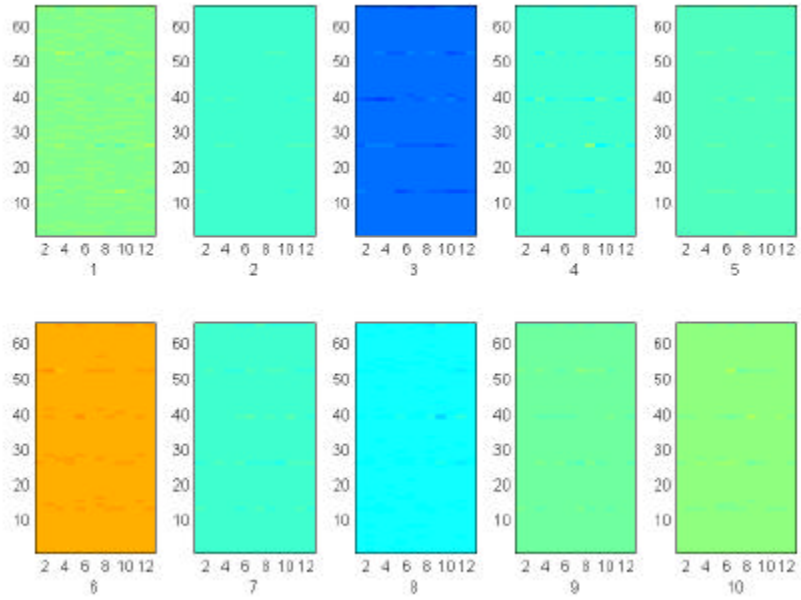


Figura 10: Matrices de Coeficientes W_i.

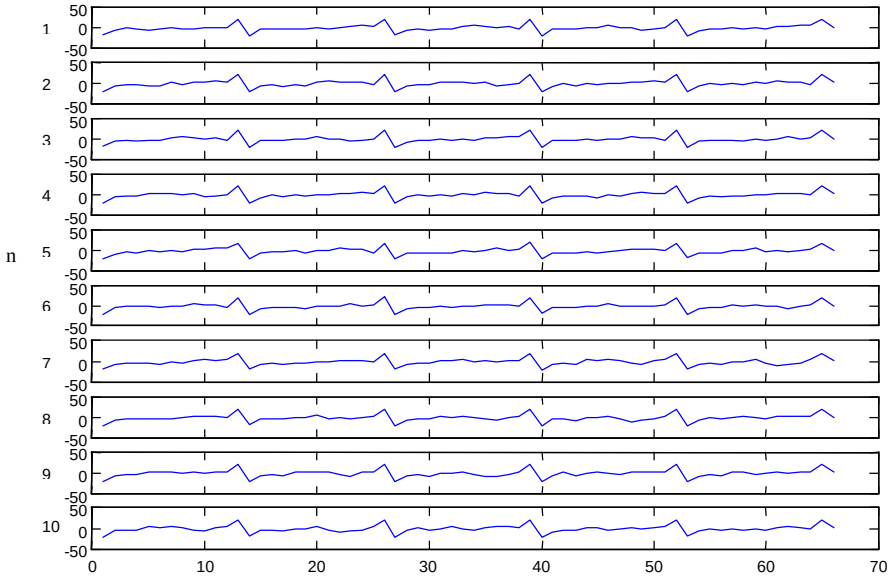


Figura 11: Algunos instantes de la linea de retardo utilizada para el filtrado Y_n.

Redes Neuronales

Las RNA intentan simular, al menos parcialmente, la estructura y funciones del cerebro y sistema nervioso de los seres vivos. Una RNA es un sistema de procesamiento de información o señales compuesto por un gran número de elementos simples de procesamiento, llamados neuronas artificiales o simplemente nodos. Dichos nodos están interconectados por uniones directas llamadas conexiones y cooperan para realizar procesamiento en paralelo con el objetivo de resolver una tarea computacional determinada [Hau95].

La aparición de técnicas de entrenamiento eficaces para redes neuronales -en particular las redes anteroalimentadas- permitió la aplicación de las mismas al procesamiento del habla, aunque hasta hace poco tiempo estuvieron orientadas a patrones estacionarios. Para evitar este escollo se diseñaron redes neuronales que - además de los patrones estáticos - incorporaran simultáneamente información que fuera generada en diferentes instantes. La utilización de redes con retardos (TDNN) permite descubrir características acústico-fonéticas y sus relaciones a lo largo

del tiempo sin verse afectadas por las traslaciones del patrón de entrada [WHH89]. Una red neuronal que no es invariante a la traslación requiere una segmentación precisa para alinear el patrón (frame) de entrada en forma adecuada. Esta invarianza al desplazamiento ha sido considerada un factor determinante en algunos modelos neuronales que han sido propuestos para reconocimiento de secuencias fonéticas y es por lo tanto muy importante para nuestra aplicación. Otra arquitectura con características similares para la clasificación de patrones dinámicos son las redes neuronales recurrentes (RNN). Existen varias simplificaciones que se pueden aplicar para extender el algoritmo de retropropagación para entrenar estas redes. Elman [Elm89], [Elm90] introdujo una clase particular de redes recurrentes en la cual existe una conexión de realimentación entre un vector de estado y la capa oculta (Figura 12). Elman utilizó esta arquitectura, junto con el algoritmo de retropropagación, para aprender la estructura gramatical de un conjunto de oraciones generadas al azar a partir de un vocabulario limitado y una gramática. El vector de estado proporcionaba la aptitud de las redes de Elman para almacenar información acerca de los estados anteriores (o salidas). En nuestro caso, si bien los patrones a clasificar son dinámicos, el hecho de utilizar patrones con información de contexto como Delta Cepstrum haría poco necesario el uso de redes neuronales dinámicas. Sin embargo esto incrementa la dimensión de los patrones complicando el espacio de búsqueda para los pesos de la red. Por otra parte POF funciona relativamente bien tratando los patrones como estáticos, esto llevó en nuestro trabajo a la utilización de una arquitectura con una dinámica sencilla como la red de Elman ya descripta .

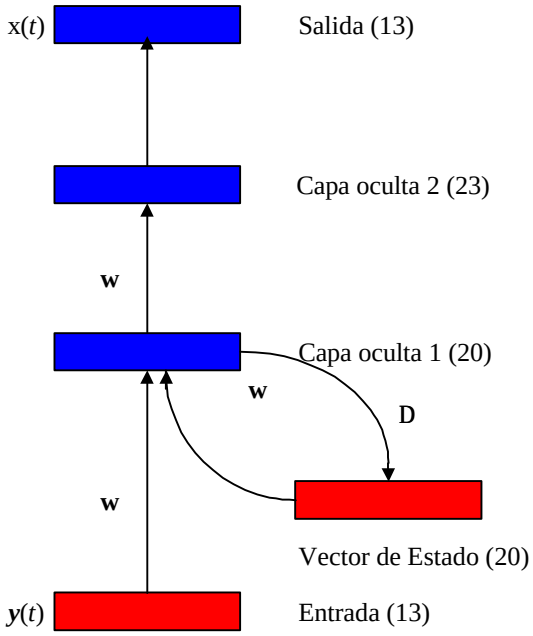


Figura 12: Arquitectura Red de Elman utilizada.

No existe un límite para fijar la cantidad de capas de este tipo de redes, pero se ha demostrado que con una capa oculta y con un número suficiente de nodos es capaz de solucionar casi cualquier problema. Si se agrega una capa oculta más, la red soluciona cualquier tipo de problemas y en forma más eficiente que con una sola capa oculta. De esta forma se puede ver a esta red como un aproximador universal de funciones con una dinámica sencilla, lo que contempla especialmente un caso no lineal, como lo es el de el mapeo del espacio de características sucio al limpio.

Para entrenar la red se utilizó el algoritmo de retropropagación según se detalla en la Tabla 5. Las entradas se normalizaron para cada dimensión del patrón en forma independiente. El entrenamiento se detuvo en el pico de generalización medido con respecto al archivo de prueba. Para más detalles se puede consultar la extensa bibliografía al respecto (por ej. [Lip87], [WiL90], [Has95]).

Tabla 5: Algoritmo de entrenamiento para la red de Elman (modo batch)

1. Inicializar pesos y umbrales en forma aleatoria.
2. En cada época:
   a. Presentar la secuencia completa de entrada a la red,
   b. Calcular sus salidas
   c. Comparar con las salidas deseadas para generar la secuencia de error.
   d. Para cada instante de tiempo retropropagar el error para encontrar los gradientes de la función error de cada peso y umbral:

$$error_i'(n) = \sum_j error_j(n) \cdot w_{i,j} \cdot y'_j(n)$$

   e. Este gradiente es solo una aproximación porque las contribuciones de los pesos y umbrales a través de la conexión recurrente retardada son ignoradas.
   f. Utilizar luego este gradiente para actualizar los pesos y umbrales en la dirección del gradiente negativo de la siguiente forma:

$$w_{i,j}(n) = w_{i,j}(n-1) + \eta \cdot error_j(n) \cdot y'_j(n) \cdot x_i(n)$$

   Donde $w_{i,j}$ es el peso que va del nodo i al nodo j, η es el coeficiente de aprendizaje, x_i es la entrada correspondiente, y'_j es la derivada de la función de activación evaluada en Σx_i y n es el instante de tiempo considerado. Para evitar los problemas de elegir un η adecuado y mejorar el desempeño del algoritmo se utiliza un coeficiente de aprendizaje adaptativo de manera de maximizar el tamaño del paso de aprendizaje pero sin perder la estabilidad.

3. Repetir hasta que se alcance el error deseado o el número de épocas prefijado.

Datos utilizados

Para los experimentos se utilizó la base de datos Latino-40, diseñada por Entropic Research (1994, Palo Alto, California) para proveer un juego de grabaciones para entrenamiento de sistemas independientes del hablante que reconozcan el español latino-americano.
El banco de datos comprende aproximadamente 5000 archivos de emisión. Estos archivos incluyen aproximadamente 125 emisiones de cada uno de 40 hablantes diferentes, 20 varones y 20 mujeres. Las oraciones fueron leídas de texto extraído de diarios latinoamericanos. Las grabaciones fueron realizadas con un micrófono de alta-calidad, montado en la cabeza (Shure SM10A) en un ambiente de oficina, y las pronunciaciones se digitalizaron en 16-bit por muestra a 16 kHz. Todos los hablantes empleados eran adultos; entre 18 y 59 años de edad. Todos declararon ser hablantes nativos de Español Latinoamericano. Siete hablantes eran de Perú; cinco de Argentina, Columbia, Guatemala, y Nicaragua; tres de Venezuela; y dos de Chile, Costa Rica, Cuba, El Salvador, y México. De dada frase se separó en segmentos de 25 mseg de duración, multiplicados por ventanas de Hamming, cada 10 mseg. De cada segmento se parametrizó con las siguientes características: 26 coeficientes, 12 MEL cepstral coeficientes, C0 coeficiente del cesptral y sus derivadas temporales.
Los datos de ruido blanco fueron tomados de la base NOISEX-92 [NOI92], los que fueron digitalizados de un generador de ruido analógico de alta calidad (Wandel & Goltermann) a 19.98 KHz y 16 bits, exhibiendo igual energía por Hz de ancho de bancha. El ruido se remuestró a 16 KHz y se mezcló con las señales de habla de LATINO-40 en distintas proporciones para lograr distintas relaciones señal ruido. Se debe aclarar que en todos los casos se ha obviado el conocido efecto Lombard (por el cual el hablante modifica en forma importante su fonación en presencia de ruido ambiente)[Lom11] [LaT71] [Jun93] y se ha procedido por simple adición de las señales, por lo que las estimaciones pueden ser sobre-optimistas en este sentido.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

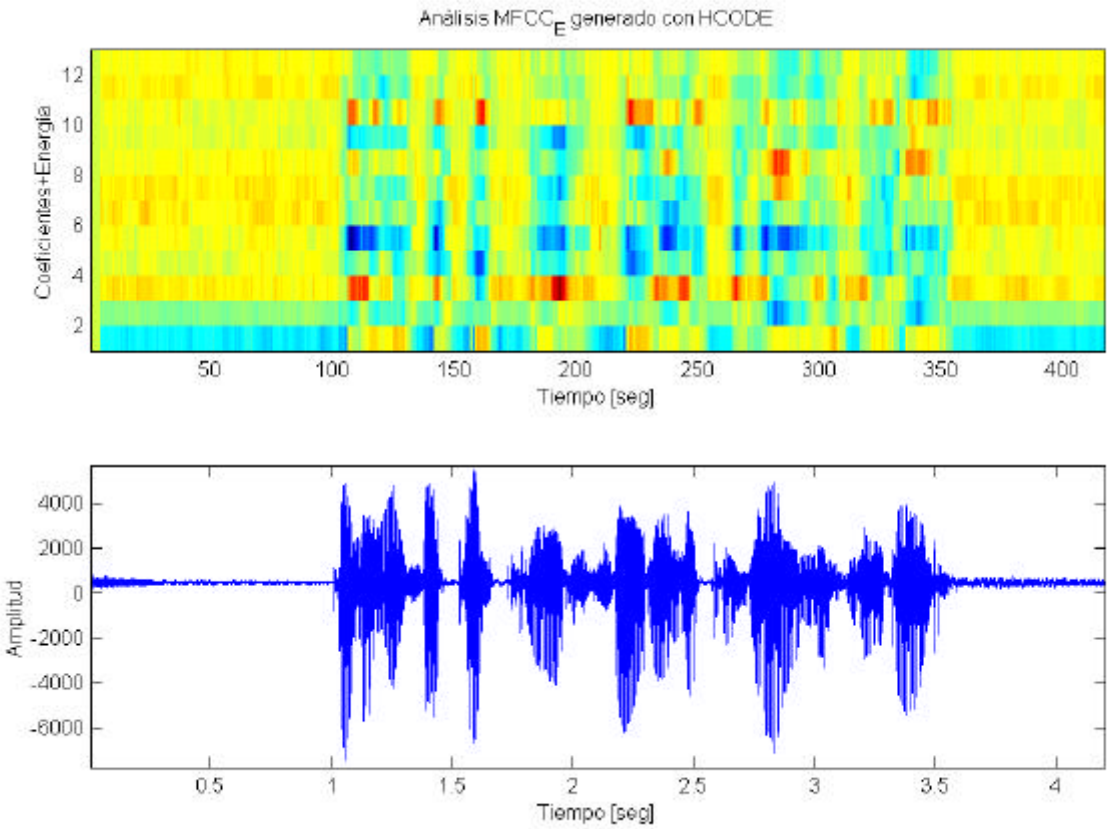


Figura 13: Ejemplo de codificación de la oración af14_001 de Latino40.

Modelos de Markov Ocultos

Para evaluar la eficacia de mapeo de los métodos planteados se utilizó un sistema sencillo de reconocimiento automático de habla continua en idioma español con independencia del hablante, midiendo dicha eficacia en términos del aumento en el porcentaje de reconocimiento por palabras (WER), con respecto al sistema de referencia. El módulo del modelo estadístico fonético-acústico de la voz está basado en modelos ocultos de Markov (HMM), [HAJ92], [You95]. En nuestro caso se utilizaron modelos continuos de mezclas gaussianas contexto-independientes para los fonemas y fue el mismo utilizado en el trabajo [GEF99]. A continuación se explica brevemente como se obtuvieron los mismos.

La primera etapa en el desarrollo fue la creación de modelos iniciales de fonemas. Se utilizaron modelos de tres estados con 50 gaussianas por estado. Para el entrenamiento de dichos modelos se utilizó la base de datos fonética inglesa TIMIT [GLF93], debido a que está etiquetada y alineada a nivel fonético. Para ello se mapearon los 64 fonemas ingleses a los 28 fonemas españoles, tomando solo los fonemas ingleses que tuvieran un equivalente fonético en español.

Una vez generados los 28 modelos iniciales se procedió a su entrenamiento con material de la base LATINO-40 (habla limpia). Con estos nuevos modelos, se generaron segmentaciones fonéticas de la base de datos de entrenamiento. Esto tiene por objetivo romper el vínculo inicial que los fonemas guardan con la fonética inglesa. A partir de estos alineamientos se reentrenaron los modelos fonéticos del español. Estos últimos modelos son los que se utilizaron en nuestras pruebas. No se utilizó modelo del lenguaje.

Para obtener los resultados del sistema de referencia se realizaron pruebas de reconocimiento sobre un conjunto de datos reducido no utilizada durante el entrenamiento (20 frases por hablante), con un vocabulario total de 677 palabras. Para el cálculo del WER se tomaron como errores solamente las exclusiones y las sustituciones de palabras.

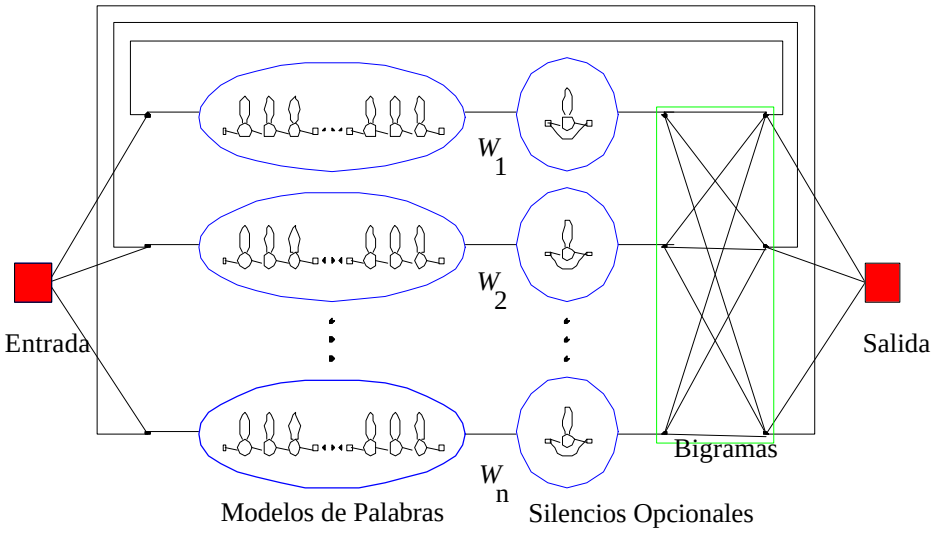


Figura 14: Esquema de un reconocedor de discurso contuuo basado en HMM.

Resultados

En esta sección presentamos los resultados de los experimentos realizados con ambos algoritmos de mapeo. Las tablas debajo muestran cuatro tipos de indicadores de desempeño:

1. *Error de reconocimiento de palabras*. (WER en %). Obtenido según se explicó en la sección anterior.
2. *Medida de distorsión relativa* (d). Para una componente dada de un vector de características definimos la distorsión relativa entre los datos limpios y ruidosos como sigue:

$$d = \sqrt{\frac{E\left[(x-y)^2\right]}{\mathrm{var}(x)}} \tag{8}$$

3. *Error Cuadrático Medio* (ECM).

$$ECM = \frac{1}{L.N}\sqrt{\sum_{n=1}^{N}\sum_{l=!}^{L}(x-y)^2} \tag{8}$$

4. *Aumento de la SNR en el espacio de la características* (ASNRC). Aquí calculamos la diferencia entre la SNR antes y después del mapeo en el espacio de las características[1], expesada en decibeles.

En las Tablas 6 y 7 se muestran los resultados finales comparativos, los que se grafican en las Figura 16 y 18. La Figura 15 muestra un ejemplo de la curva de aprendizaje de NNNLF. Todos los algoritmos se implementaron en lenguaje Matlab v5.2, el código fuente se muestra en el Apéndice A. En la Figura 18 se muestran los resultados de la salida del intérprete de comandos del Matlab luego de invocar al programa *POFTrain.m.* El reconocedor basado en HMM's se implementó mediante scripts en lenguaje CShell para el HTK v3.0. En la Figura 20 se muestra una comparación entre los vectores de características generados por cada método.

Tabla 6: Porcentaje de Reconocimiento de Palabras (WER %)

	Limpios	100 dB	30 dB	20 dB	10 dB
Referencia	68,90	61,58	16,07	7,65	6,78
POF	-	61,04	37,8	17,54	11,27
NNNLF	-	12,20	8,36	7,76	10,53

[1] No confundir con la SNR a la cual se mezclaron los datos sucios y limpios, que está calculada en el dominio del tiempo.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

Tabla 7: Otros indicadores de Desempeño

Resultado	Mapeo / SNR	100 dB	30 dB	20 dB	10 dB
ECM	**POF**	0,0566	0,0667	0,0699	0,0577
	NNNLF	0,0741	0,0720	0,0754	0,0682
GSNR	**POF**	-6,3462	0,5451	1,3990	4,0651
	NNNLF	-8,6790	-0,1130	0,7366	2,3452
d (Promedio)	**POF**	0,3169	0,8129	0,9586	1,0800
	NNNLF	0,3169	0,8133	0,9586	1,0580
Tiempo	**POF**	8,7840	8,7284	8,8333	8,7300
	NNNLF	14,6009	14,2479	11,8411	11,6321

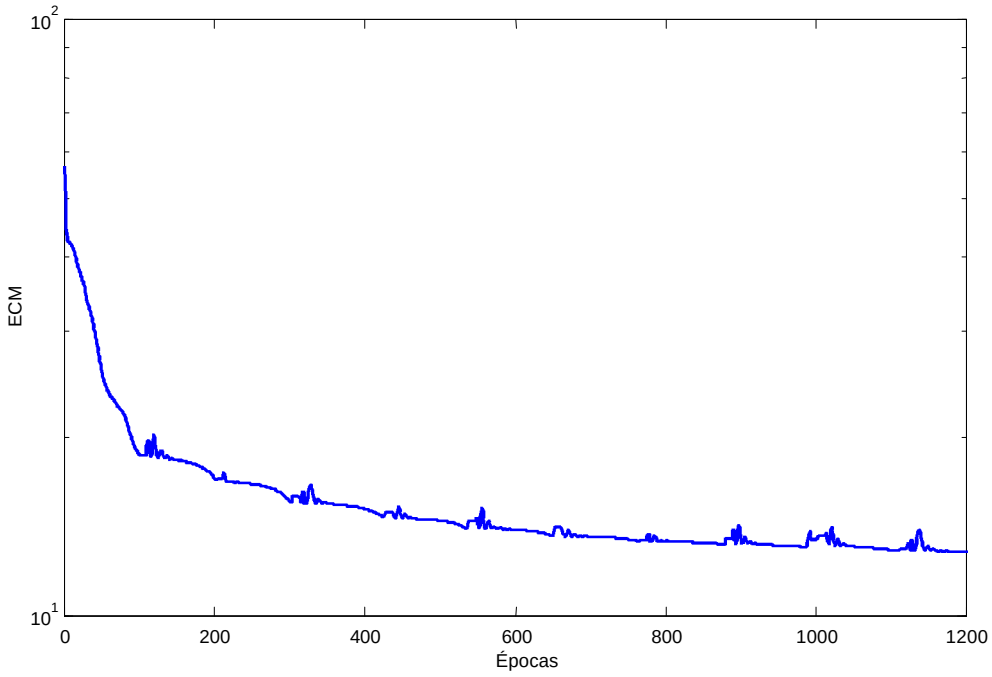


Figura 15: Evolución del ECM para NNNLF entrenado sobre 10 emisiones, SNR 20 dB.

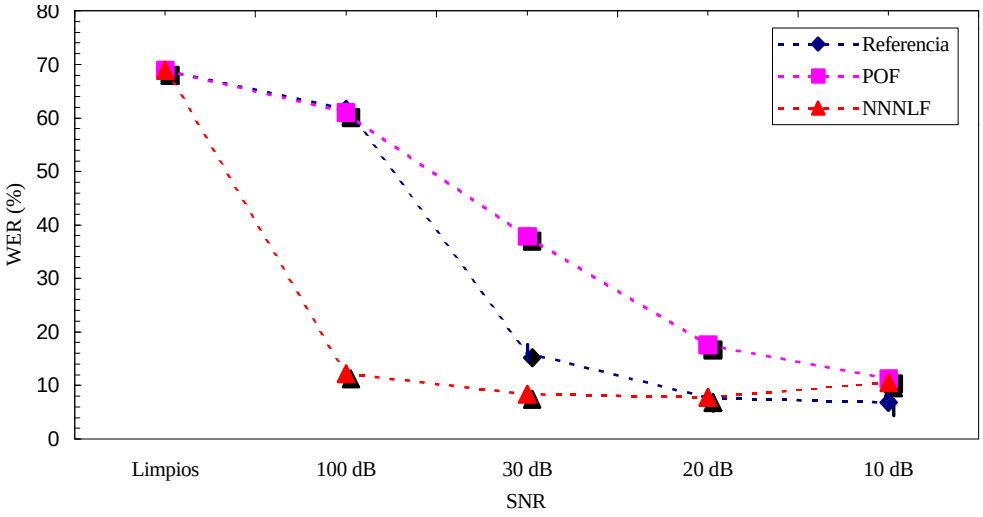


Figura 16: Resultados WER.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

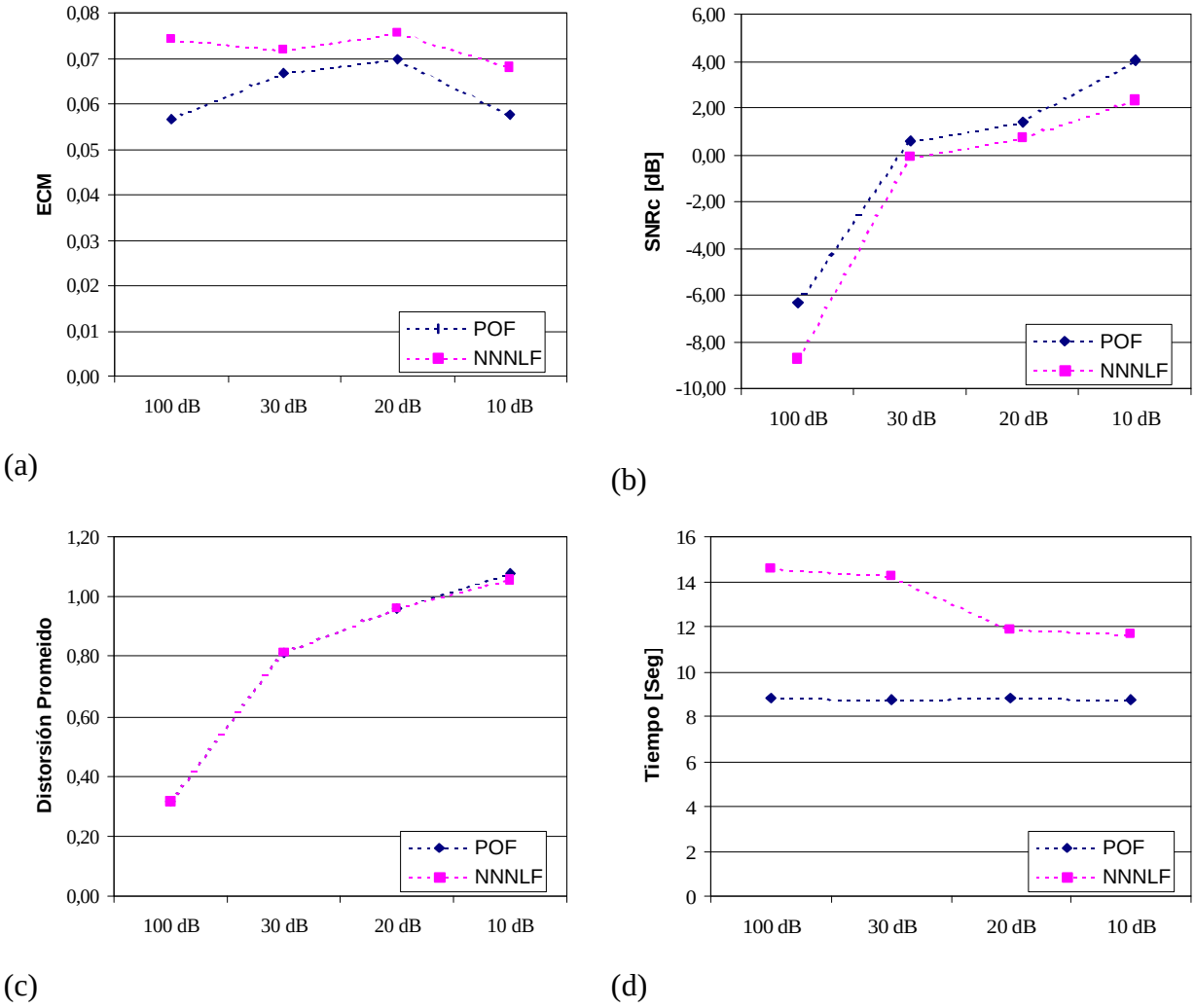


Figura 17: Desempeño Relativo en función de la SNR: (a) Error cCuadrático Medio, (b) Ganancia SNR (en el dominio de las características), (c) Distorsión promedio, (d) Tiempo de corrida.

```
POF Train (Filtrado óptimo probabilistico)
-------------------------------------------
P=2, I=10, epochs VQ=1500
Leyendo archivos de datos...
Leyendo datos origen
Hablante: af14_001.mfc
El tipo de datos es MFCC_E
Leyendo datos destino
Hablante: af14_001.mfc
El tipo de datos es MFCC_E
Realizando una VQ de los datos destino...
Entrenando el cuantizador...
TRAINWB1, Maximum epoch reached.
Grabando parametros del cuantizador...
Cuantizando...
Calculando probabilidad que un vector pertenezca al cluster g(i) (Pg(i))...
Calculando parámetros P(zn|gi) como mezcla Gaussiana...
Calculando P(zn)...
Calculando P(gi|zn)...
Calculando matrices de correlación...
Calculando coeficientes del filtro...
Grabando parametros del filtro...

elapsed_time =   79.42
```

Figura 18: Salida del programa de entrenamiento de POF

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
H. L. Rufiner, C. E. Martínez & H. M. Torres; "Comparación entre Filtrado Óptimo Probabilístico y Filtrado no lineal mediante Redes Neuronales para limpieza de habla continua con ruido"
Anales de las VIII Jornadas de Jóvenes Investigadores Grupo Montevideo, pp. 76, Sep, 2000.

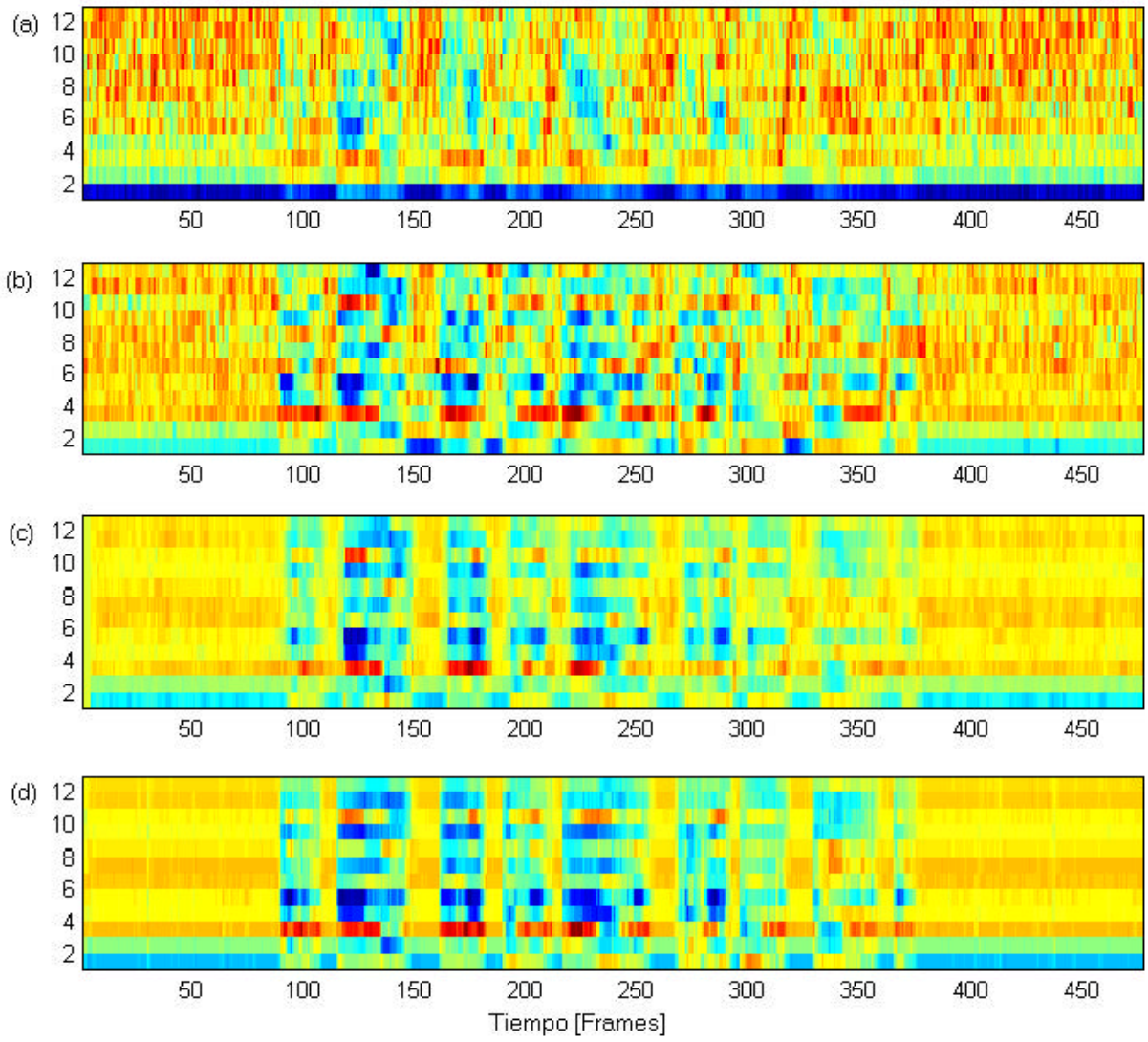


Figura 19: Ejemplo de resultados sobre la emisión af14_001 luego de ser entrenados ambos métodos sobre 10 emisiones de la base de datos: (a) señal sucia con ruido blanco a 10 dB, (b) señal limpia x, (c) señal limpiada con POF, (d) señal limpiada con NNNLF (1200 épocas).

Discusión y Conclusiones

Hemos presentado y discutido dos algoritmos de mapeo de características capaces de aprovechar las relaciones no lineales entre dos espacios acústicos. Esto permite mejorar la actuación del reconocedor en la presencia de un señal ruidosa usando una base de datos pequeña con grabaciones simultáneas en los ambientes acústicos limpios y ruidosos.

El algoritmo de mapeo POF se ha comportado bien en una tarea sencilla de reconocimiento independiente del hablante en idioma castellano con ruido aditivo estacionario. POF aprovechó eficazmente las correlaciones dentro y entre los frames, produciendo mejoras importantes por encima del sistema sin mapeo.

Como vimos, las ANN permiten el mapeo directo entre ambos espacios de manera no lineal lo que constituiría un verdadero filtro no lineal. Sin embargo los resultados a nivel WER obtenidos con NNNLF no han sido tan satisfactorios, a pesar de que en todos los casos el ECM es menor que para POF, los resultados son siempre inferiores (y hasta peores que la referencia). Esto puede indicar que si bien el error es menor, NNNLF elimina el ruido a costa de distorsionar de alguna manera la señal limpia. Esto puede indicar que el criterio utilizado para guiar el entrenamiento no es el adecuado, teniendo en cuenta la naturaleza probabilística del reconocedor

como se plantea en [YChF97], [Yuk99]. Tampoco se evidenció la ventaja de incluir la componente dinámica en el mapeo. Por otra parte los procedimientos utilizados para entrenar las redes resultaron excesivamente lentos y muy propensos a caer en mínimos locales, cosa que pudo constatarse en los experimentos luego de realizar varias corridas. El tiempo típico de entrenamiento de NNNLF con el equipo utilizado en las pruebas (PC Celeron 333MHz) fue de unas 40 horas (para 1200 épocas), contra una media hora de POF. Este último inconveniente hizo bastante engorroso el ensayo de distintas configuraciones de la red, por lo cual es posible que con más experimentos se logré encontrar una que mejor se adapte al problema. Como ya se ha visto en otras aplicaciones la capacidad teórica de las redes puede ser universal, pero debe resolverse adecuadamente la forma de encontrar los parámetros óptimos para la aplicación considerada en tiempos razonables. Como continuación del presente trabajo deberían ensayarse otras posibles arquitecturas de redes neuronales dinámicas y estáticas, así como evaluar la posibilidad de emplear métodos de aprendizaje menos propensos a caer en mínimos locales. Otra línea a explorar podría ser la explotar la relación entre los NNNLF y POF que surge de ver a las redes como estimadores de funciones de densidad de probabilidad.

Referencias

[Ace90] A. Acero, "Acoustical and Environmental Robustness in Automatic Speech Recognition", Ph.D. Thesis, CarnegieMellon University, September 1990.

[DiM94] V. Digalakis, H. Murveit, "GENONES: Optimizing the Degree of Mixture Tying in a Large Vocabulary Hidden Markov Model Based Speech Recognizer", 1994 IEEE ICASSP.

[Dod92] G. Doddington, "CSR Corpus Development", 1992 DARPA SLS Workshop, pp. 363-366.

[Elm90] J.L. Elman. “Finding structure in time”. *Cognitive Science* 14 (1990) 179-211.

[ErW89] A. Erell, M. Weintraub, "Spectral Estimation for Noise Robust Speech Recognition", 1989 DARPA SLS Workshop, pp. 319-324.

[ErW90] A. Erell, M. Weintraub, "Recognition of Noisy Speech: Using Minimum-Mean Log-Spectral Distance Estimation", 1990 DARPA SLS Workshop, pp. 341-345.

[Fur81] S.F. Furui, "Cepstral Analysis Technique for Automatic Speaker Verification", IEEE Trans. ASSP, Vol. 29, pp. 254-272, April 1981.

[GCR90] H. Gish, YL. Chow, J.R. Rohlicek, "Probabilistic Vector Mapping of Noisy Speech Parameters for HMM Word Spotting", 1990 IEEE ICASSP, pp. 117-120.

[GEF99] Graciarena M., Estienne C., Ferrer L., “Reconocimiento Automático de Habla Continua en Español sobre un Vocabulario Restringido”, Anales de la VIII Reunión de Trabajo en Procesamiento de la Información y Control (RPIC 99), Mar del Plata, Argentina, 23-25 Sep. 1999.

[GLF93] Garofolo, Lamel, Fisher, Fiscus, Pallett, Dahlgren, “*DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus Documentation”*. National Institute of Standards and Technology. February 1993.

[HAJ92] Huang, X. D., Ariki, Y, Jack, ,M, *Hidden Markov Models for Speech Recogniton*, Edinburgh University Press, (1992)

[Hau95] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks.* The MIT Press. 1995.

[Jun93] Junqua J.C., “The Lombard reflex and its role on human listeners and automatic speech recognizers”, J.Acoust.Soc.Amer.,93-1, pp. 637-642 (1993).

[JuR87] B.H. Juang, L.R. Rabiner, "Signal Restoration by Spectral Mapping", 1987 IEEE ICASSP, pp. 2368-2371.

[LaT71] Lane H.L., Tranel B., “The Lombard sign and the role on hearing in speech”, Journal of Speech and Hearing Research, Vol. 14, pp. 677-709 (1971).

[LBG80] Y. Linde, A. Buzo, R.M. Gray, "An Algorithm for Vector Quantizer Design", IEEE Trans. Comm., vol. 28, pp. 84-95, January 1980.

[Llo82] S. P. Lloyd., “Least squares quantization in pcm”, *IEEE Transactions on Information Theory*, Nº 28 (IT), pp 127-135, Marzo 1982.

[Lom11] Lombard E., “Le signe de l’elevation de la voix”, Annales Maladies Oreilles, Larynx, Nez, Pharynx, Vol 37 pp 101-119, 1911.

[MBD93] H. Murveit, J. Butzberger, V. Digalakis, M. Weintraub, "Large Vocabulary Dictation Using SRI's DECIPHER Speech Recognition System: Progressive Search Techniques", 1993 IEEE ICASSP. pp. 11319-11322.

[MBW92] H. Murveit, J. Butzberger, M. Weintraub, "Performance of SRI's DECIPHER™ Speech Recognition System on DARPA's CSR Task", 1992 DARPA Speech and Natural Language Workshop Proceedings, pp. 410-414.

[MBW92b] Murveit, H., J. Butzberger, M. Weintraub, "Reduced Channel Dependence for Speech Recognition", 1992 DARPA Speech and Natural Language Workshop Proceedings, pp. 280-284.

[NeW94] L. Neumeyer, M. Weintraub, "Probabilistic Optimum Filtering for Robust Speech Recognition", 1994 IEEE- ICASSP, pp. 417-420.
[NGR92]K. Ng, H. Gish, J.R. Rohlicek, "Robust Mapping of Noisy Speech Parameters for HMM. Word Spotting", 1992 IEEE ICASSP, pp. II-109-II-112.
[NNP88] A. Nadas, D. Nahamoo, M. Picheny, "Adaptive Labeling: Normalization of Speech by Adaptive Transformations based on Vector Quantization", 1988 IEEE-ICASSP, pp. 521-524.
[NOI92] NOISEX-92, Base de Datos con varios tipos de ruido disponible por ftp anónimo en *IEEE's Signal Processing Information Base*.
[RaJ93] Rabiner L R.., Juang B., *Fundamentals of Speech Recognition*. Prentice Hall, (1993)
[RaS78] Rabiner L R.and Schafer R. W., *Digital Processing of Speech Signals*. Englewood Cliffs, N.J.: Prentice Hall, (1978).
[San96] A. Sankar, "A Maxumum-Likelihooh Approach to Stochastic Matching for Robust Speech Recognition", IEEE Trans. On Speech and Audio Proc., Vol 4, Nº 3, pp 190-201, Mayo 1996.
[SLO92] R.M. Stern, F.H. Leu, Y. Ohshima, T.M. Sullivan, A. Acero, "Multiple Approaches to Robust Speech Recognition", 1992 International Conference on Spoken Language Processing, pp. 695-698.
[WeN94] M. Weintraub, L. Neumeyer, "Constructing Telephone Acoustic Models from a High-Quality Speech Corpus", 1994 IEEE ICASSP.
[WHH89] A. Waibel., T. Hanazawa., G. Hinton., K. Shikano., K. Lang; "Phoneme Recognition Using Time-Delay Neural Networks". *IEEE Trans. ASSP* Vol. 37. No 3 (1989).
[YChF97] Yuk D., Che Ch., Flanagan J., "Robust Speech Recognition using Maximum Likelihood Neural Networks and Continuous Density Hidden Markov Models, IEEE Workshop on Automatic Speech Recognition and Understanding", pp. 474 - 481, Diciembre 14 - 17, 1997, Santa Babara, California.
[You95] Young S. "Large Vocabulary Continuous Speech Recognition: A review". *IEEE Workshop on Speech Recogniton*, Diciembre (1995).
[Yuk99] Yuk D. , "Robust Speech Recognition using Neural Networks and Hidden Markov Models - Adaptations using Non-Linear Transformations -", PhD Thesis, The State University of New Jersey, Graduate Program in Department of Computer Science, New Brunswick, New Jersey, Octubre, 1999.

Apéndice A

Código Fuente de la implementación de POF en lenguaje Matlab v5.2 o posterior

```
%POFTrain
%Entrena un Filtro óptimo probabilistico y guarda los parámetros en un archivo
%Una vez entrenado el filtrado se realiza con PofRun.m
%
%TOOLBOX NECESARIOS: Signal Processing, Neural Netoworks.
%MATLAB VERSION: 5.3
%Autor: HLR, Abril 2000

%Este método se puede usar como mapeo de los datos con ruido a limpios o viceversa

disp('POF Train (Filtrado óptimo probabilistico)');
disp('-----------------------------------------');
clear all
close all

tic

grafica=1;  %Activa o desactiva salida gráfica
P=2;        %orden del filtro
%-------------------------------------------------------------------------------
%Realizo una VQ de los datos limpios

%Leo los datos Limpios xn y Sucios yn
disp('Leyendo archivos de datos...');
[x,y,N,L,sampKind]=CargaDatosxy;
```

```
%Realizo una cuantización vectorial (VQ) de los archivos de características de destino
%con método de aprendizaje competitivo autoorganizado-----------------------------------
disp('Realizando una VQ de los datos destino...');

if ~exist('DatosVQ.mat')
   disp('Entrenando el cuantizador...');

   I=10;          %Cantidad de clusters o grupos de vectores
   epochs=1500;     %Épocas de entrenamiento

   %Creo una capa competitiva con I neuronas (UNA POR CLUSTER)inicializadas entre 0 y 1
   net=newc([zeros(L,1) ones(L,1)],I);

   %Entreno la red durante una determinada cantidad de épocas
   net.trainParam.epochs=epochs;
   net=train(net,x);

   disp('Grabando parametros del cuantizador...');
   save 'DatosVQ.mat' net I
else
   disp('Leyendo parametros del cuantizador...');
   load 'DatosVQ.mat'
end

disp('Cuantizando...');

%Calculo las salidas o activaciones de la red
act=sim(net,x);

%Le asigno a cada entrada su indice de cluster correspondiente de acuerdo a la salida de la red
vq=vec2ind(act);
act=full(act);
```

```
if grafica,
   figure
%Grafico los resultados de la clasificación en clusters
   imagesc(act);
   title('Resultado de la cuantización vectorial (VQ)');
   xlabel('Patrones');
   ylabel('Cluster');
end,

%-------------------------------------------------------------------------------
%Calculo la probabilidad de que un vector pertenezca a un cluster determinado Pg(i)
%o probabilidad a priori
disp('Calculando probabilidad que un vector pertenezca al cluster g(i) (Pg(i))...');

for i=1:I;
   n=length(find(vq==i));
   Pg(i)=n/N;
end;

%Gráfico la pdf
if grafica,
   figure
   bar(Pg);
   title('Probabilidad que un vector pertenezca al cluster g_i');
   xlabel('Cluster (g_i)');
   ylabel('P(g_i)');
end,

%-------------------------------------------------------------------------------
%Calculo los parámetros de P(zn|gi) (likelihood) con la version ruidosa de los datos yn
%(con una pdf gaussiana por cluster)
%En esta version zn=yn (caracteristica ruidosa)

%Recorro los patrones y calculo los parámetros de las Gaussianas de cada cluster
```

```
%Limpio el cluster auxiliar
Gi=[];
disp('Calculando parámetros P(zn|gi) como mezcla Gaussiana...');
%Inicializo variables
mu=[];              %Medias
ejs=8;              %Cantidad de ejemplos mostrados en la gráfica por cluster

if grafica,
   figure('name','Ejemplos de vectores sucios y Medias de los clusters');
end,


%Para cada cluster...
for i=1:I,
   gi=find(vq==i);       %Encuentro los vectores que pertenecen al cluster
   Gi=y(:,[gi]);         %Los pongo aparte en un vector aux
   G(i)={Gi};            %Los separo en un cell array
   %Cálculo los parámetros de la pdf del cluster
   if ~isempty(Gi),
      mu=[mu; mean(Gi')];
   else
      mu=zeros(L,1);
   end,


   %Los guardo en un arreglo
   if i==1,
      C=cov(Gi');
   else
      aux=cov(Gi');
      C=cat(3, C, aux);
   end,

   if grafica,
```

```
            %Grafico ejemplos de vectores del clusters
            h=subplot(ejs,I,(n-1)*I+i);
         for n=1:ejs,
            plot(y(:,gi(n)));
            %axis off
            %h=get(h,'CurrentAxes');
            set(h,'FontSize',7)
            set(h,'XTick',[1 6 13])
            set(h,'YTick',[])
         end,
         %Grafico la media de cada cluster
         h=subplot(ejs,I,(8-1)*I+i);
         plot(mean(Gi'),'r');
         set(h,'FontSize',7)
         set(h,'XTick',[1 6 13])
         set(h,'YTick',[])
         legend('Media','Vectores')
      end,
   end,

   if grafica,
      figure('name','Matrices de covarianza, Medias y varianzas de los clusters');

      %Grafica Matrices de covarianza de los clusters
      for i=1:I,
         h=subplot(4,I/2,i);
         pcolor(C(:,:,i));
         set(h,'FontSize',7)
         axis ij
         title(int2str(i),'Color','b');
      end;


      %Grafica Medias y varianzas de los clusters
```

```
   legend('Media','Varianza')
   for i=1:I,
end,

%-------------------------------------------------------------------------------
%Ahora calculo P(zn) como:
%P(zn)=sum(P(zn|gi)*P(gi))
%       I
%En esta version zn=yn (caracteristica ruidosa)
disp('Calculando P(zn)...');
Pz=zeros(1,N);
Pzg=zeros(N,I);
for n=1:N,
   for i=1:I,
      Pzg(n,i)=normdist(y(:,n)',mu(i,:),C(:,:,i));
      Pz(n)=Pz(n)+Pzg(n,i)*Pg(i);
   end,
end,

if grafica,
   figure
   bar(Pz);
   title('Probabilidad de ocurrencia de los vectores sucios P(z_n)');
   ylabel('P(z_n)')
   xlabel('_n')
end,

%-------------------------------------------------------------------------------
%Ahora calculo P(gi|zn) usando Bayes como:
%P(gi|zn)=P(zn|gi)*P(gi)/P(zn)
disp('Calculando P(gi|zn)...');
Pgz=zeros(I,N);
```

```
   for i=1:I,
      Pgz(i,n)=Pzg(n,i)*Pg(i)/Pz(n);
   end;
end;

if grafica,
   figure
   pcolor(Pgz)
   shading flat
   title('P(g_i|z_n)');
   xlabel('_n')
   ylabel('_i')

   figure
   pcolor(Pzg)
   shading flat
   title('P(z_n|g_i)');
   xlabel('_i')
   ylabel('_n')
end,

%-------------------------------------------------------------------------------
%Calculo las matrices de correlacion
disp('Calculando matrices de correlación...');

%Armo la linea de retardo de los vectores sucios (Y)
%Y=[];
Y=[reshape(y(:,1:2*P+1),(2*P+1)*L,1)' 1]';
for n=P+2:N-1-P+1,
   Y=cat(2,Y,[reshape(y(:,n-P:n+P),(2*P+1)*L,1)' 1]');
end,

%Grafico algunas lineas
```

```
   figure('name','Linea de retardo para el filtrado');
   for n=1:10,
      subplot(10,1,n)
      plot(Y(:,n)');
      ylabel(int2str(n))
   end;
end,
%Calculo las matrices y vectores de correlacion
for i=1:I,
   aux=zeros((2*P+1)*L+1,(2*P+1)*L+1);
   for n=P+1:N-1-P+1;
      aux=aux+Y(:,n-P)*Y(:,n-P)'*Pgz(i,n);
   end,
   if i==1,
      R=aux;
   else
      R=cat(3,R,aux);
   end
end,

for i=1:I,
   aux=zeros((2*P+1)*L+1,L);
   for n=P+1:N-1-P+1;
      aux=aux+Y(:,n-P)*x(:,n)'*Pgz(i,n);
   end,
   if i==1,
      r=aux;
   else
      r=cat(3,r,aux);
   end
end,

if grafica,
```

```
    for i=1:I,
       subplot(2,I/2,i);
    figure('name','Matrices de Autocorrelacion R');
       pcolor(R(:,:,i));
       xlabel(int2str(i));
       shading flat
    end;

    figure('name','Matrices de Correlacion Cruzada r');
    for i=1:I,
       subplot(2,I/2,i);
       pcolor(r(:,:,i));
       xlabel(int2str(i));
       shading flat
    end;
end,
%-------------------------------------------------------------------------------
%Calculo los coeficientes del filtro
disp('Calculando coeficientes del filtro...');

W=[];
for i=1:I,
   W(:,:,i)=R(:,:,i)^-1*r(:,:,i);
end,

if grafica,
   figure('name','Matrices de Coeficientes W');
   for i=1:I,
      subplot(2,I/2,i);
      pcolor(W(:,:,i));
      xlabel(int2str(i));
      shading flat
   end;
end,
```

```
%-------------------------------------------------------------------------------
%Crabo los parametros del filtro, las distribuciones y del cuantizador
disp('Grabando parametros del filtro...');
save 'pofparams' mu C W P I Pg

%-------------------------------------------------------------------------------
toc

%EOF POFTRAIN.M
```

```
%POFRun
%Realiza un Filtrado óptimo probabilistico
%Autor: HLR, Mayo 2000
%TOOLBOX NECESARIOS: Signal Processing, Neural Networks.
%MATLAB VERSION: 5.3
%Autor: HLR, Abril 2000

%Este método se puede usar como mapeo de los datos con ruido a limpios o viceversa
%Se entrena mediante PofTrain.m

disp('POF (Filtrado óptimo probabilistico)');
disp('-----------------------------------');

path='\\AZULYORO\lrufine\Mis_documentos\Denoising\Datos\';
datosx='Limpios\caract\mfcc.logE\'; %directorio datos origen
datosxest='Pof\caract\mfcc.logE\'; %directorio datos pof
datosy='Sucios\snr10db\caract\mfcc.logE\'; %directorio datos origen
speaker='af14_001.mfc';
params='pofparams'

tic

grafica=1;  %Activa o desactiva salida gráfica
%-------------------------------------------------------------------------------
%Leo los datos origen
disp('Leyendo archivos de datos origen...');
[y,N,sampPeriod,sampSize,sampKind]=LeeHTK([path datosy speaker]);

%-------------------------------------------------------------------------------
%Leo los datos destino (para comparar)
disp('Leyendo archivos de datos destino para comparar...');
[x,N,sampPeriod,sampSize,sampKind]=LeeHTK([path datosx speaker]);
```

```
%dividido el tamaño de cada valor (real de cuato bytes o float32)
L=sampSize/4;
%La dimensión de los vectores de caracteristicas es el tamaño total de la muestra

%-------------------------------------------------------------------------------
%Cargo los parametros del filtro
disp('Cargando los parametros del filtro...');
load 'pofparams'
disp(['Orden del filtro...:' int2str(P)]);
disp(['Cantidad de Clusters: ' int2str(I)]);

%-------------------------------------------------------------------------------
%Ahora calculo P(zn) como:
%P(zn)=sum(P(zn|gi)*P(gi))
%y
%P(zn|gi) con los parámetros de las gaussianas
%En esta version zn=yn (caracteristica ruidosa)
disp('Calculando P(zn)...');
Pz=zeros(1,N);
Pzg=zeros(N,I);
for n=1:N,
   for i=1:I,
      Pzg(n,i)=normdist(y(:,n)',mu(i,:),C(:,:,i));
      Pz(n)=Pz(n)+Pzg(n,i)*Pg(i);
   end,
end,

%-------------------------------------------------------------------------------
%Ahora calculo P(gi|zn) usando Bayes como:
%P(gi|zn)=P(zn|gi)*P(gi)/P(zn)
disp('Calculando P(gi|zn)...');
Pgz=zeros(I,N);
for n=1:N,
   for i=1:I,
```

```
   end;
end;  Pgz(i,n)=Pzg(n,i)*Pg(i)/Pz(n);

if grafica,
   figure
   pcolor(Pgz)
   shading flat
   title('P(g_i|z_n)');
   xlabel('_n')
   ylabel('_i')

   figure
   pcolor(Pzg)
   shading flat
   title('P(z_n|g_i)');
   xlabel('_i')
   ylabel('_n')
end,

%-------------------------------------------------------------------------------
%Armo la linea de retardo de los vectores sucios (Y)
%Y=[];
disp('Armando la linea de retardo de los vectores sucios (Y)...');
Y=[reshape(y(:,1:2*P+1),(2*P+1)*L,1)' 1]';
for n=P+2:N-1-P+1,
   Y=cat(2,Y,[reshape(y(:,n-P:n+P),(2*P+1)*L,1)' 1]');
end,

%-------------------------------------------------------------------------------
%Ahora calculo el vector de características limpio estimado integrando las salidas de todos
%los filtros
disp('Calculando vector limpio estimado...');
xest=zeros(L,2*P); %agrego columnas vacias al principio para igualar el tamanio con x
```

```
for n=1:N-2*P,
   aux=zeros(L,1);
   for i=1:I,
      aux=aux+W(:,:,i)'*Y(:,n)*Pgz(i,n);
   end,
   xest=cat(2,xest,aux);
end;

%Grabo los resultados en un nuevo archivo
GrabaHTK([path datosxest speaker],xest,N,sampPeriod,sampSize,sampKind);

%Grafico resultados
if grafica,
   figure
   subplot(3,1,1);
   pcolor(y);
   xlabel('y: vector sucio');
   shading flat
   title('Resultados');

   subplot(3,1,2);
   pcolor(x);
   xlabel('x: vector limpio');
   shading flat

   subplot(3,1,3);
   pcolor(xest);
   xlabel('xest: vector limpio estimado');
   shading flat
end,
%-------------------------------------------------------------------------------
%Calculo algunos indicadores de desempeño
```

```
ECM=sqrt(sum(sum((xest-x).^2)))/(L*N)
%Calculo el ECM
%Calculo SNR
ruido=y-x;
ruidopof=xest-x;
%Energía del ruido original
eruido=sum(sum(ruido.^2));

%Energía del ruido del proceso de denoissing
eruidopof=sum(sum(ruidopof.^2));
%Energía de la senial
esenial=sum(sum(x.^2));

snrpof=10*log10(esenial/eruidopof)
snroriginal=10*log10(esenial/eruido)

%Calculo la medida de distorsion relativa
d=sqrt(mean(ruido'.^2)./var(x'))

toc
%EOF POFRUN.M
```

```
%NNNLFTrain
%Enrena un Filtro No lineal mediante Redes Neuronales de Elman
%Autor: HLR, Mayo 2000
%TOOLBOX NECESARIOS: Signal Processing, Neural Networks.
%MATLAB VERSION: 5.3
%Autor: HLR, Abril 2000

I=10 %equivalente al numero de clusters
grafica=1;

%Cargo vectores de patrones y objetivos para entrenar
[x,y,N,L,TipoDatos]=CargaDatosxy;

%Normaliza entrada
[yn,miny,maxy] = premnmx(y);

%Normaliza salida
xn=x;minx=0;maxx=0;

%Los convierto en secuencias
yseq = con2seq(yn);
xseq = con2seq(xn);

%Creo la red de Elman
net = newelm([zeros(L,1) ones(L,1)],[L 2*I I+L L],{'tansig','tansig','tansig','purelin'});

tic

%La entreno
net.trainParam.epochs=1200;
net.trainParam.mu=0.5;
net.trainParam.show=1;
```

```
toc
net = train(net,yseq,xseq);
%Calculo la salida
xest = sim(net,yseq);
xest = seq2con(xest);
xest = xest{:};

%Grafico resultados
if grafica,
   figure
   subplot(3,1,1);
   pcolor(y);
   xlabel('y: vector sucio');
   shading flat
   title('Resultados');
   %colorbar;

   subplot(3,1,2);
   pcolor(x);
   xlabel('x: vector limpio');
   shading flat
   %colorbar;

   subplot(3,1,3);
   pcolor(xest);
   xlabel('xest: vector limpio estimado');
   shading flat
   %colorbar;
end,
%-------------------------------------------------------------------------------
%Calculo algunos indicadores de desempeño

%Calculo el ECM
```

```
%Calculo SNR
ECM=sqrt(sum(sum((xest-x).^2)))/(L*N)
ruido=y-x;
ruidonnnlf=xest-x;
%Energía del ruido original
eruido=sum(sum(ruido.^2));
%Energía del ruido del proceso de denoissing
eruidonnnlf=sum(sum(ruidonnnlf.^2));
%Energía de la senial
esenial=sum(sum(x.^2));

snrnnnlf=10*log10(esenial/eruidonnnlf)
snroriginal=10*log10(esenial/eruido)

%Calculo la medida de distorsion relativa
d=sqrt(mean(ruido'.^2)./var(x'))

%Guardo la red entrenada..
save 'redelman1200_100db.mat' net I L N miny maxy minx maxx

%EOF NNNLFTRAIN.M
function NNNLFRun(Lista, params)

diary('elman20db.log') %Archivo de registro
date
disp('ELMANPOF (Filtrado No Lineal Dinámico)');
disp('-------------------------------------');

grafica=0;

if nargin < 1
   Origen=cellstr('E:\Datos\LATINO40\Denoissing\Sucios\snr30db\caract\mfcc.logE\af14_001.mfc');
   Destino=cellstr('E:\Datos\LATINO40\Denoissing\NLPof\caract\mfcc.logE\af14_001.mfc');
```

```
   params='pofparams'
   grafica=1;
   Deseado=cellstr('E:\Datos\LATINO40\Denoissing\Limpios\caract\mfcc.logE\af14_001.mfc');
else
   [Origen,Destino,Deseado]=textread(Lista,'%s %s %s\n');
end;
renglones=length(Origen);

I=10 %equivalente al numero de clusters
L=13 %habria que leerla de los datos

%Cargo los parametros de la red entrenada
load 'redelman1200_20db.mat'

for r=1:renglones;
   tic
   %Muestro por donde voy
   disp([char(Origen(r)) '->' char(Destino(r))]);

   %-------------------------------------------------------------------------------
   %Leo los datos origen (sucios)
   disp('Leyendo archivos de datos origen...');
   [y,N,sampPeriod,sampSize,sampKind]=LeeHTK(char(Origen(r)));

   %-------------------------------------------------------------------------------
   %Leo los datos destino (para comparar)
   disp('Leyendo archivos de datos destino para comparar...');
   [x,N,sampPeriod,sampSize,sampKind]=LeeHTK(char(Deseado(r)));

   %La dimensión de los vectores de caracteristicas es el tamaño total de la muestra
   %dividido el tamaño de cada valor (real de cuato bytes o float32)
   L=sampSize/4;

   %Normaliza entrada (con los valores extraidos del archivo)
```

```
%-------------------------------------------------------------------------
yn = tramnmx(y,miny,maxy);
%Los convierto en secuencias
yseq = con2seq(y);

%Calculo la salida
xestn = sim(net,yseq);

%Desnormaliza salida
xest = xestn;

xest = seq2con(xest);
xest = xest{:};

%Grabo los resultados en un nuevo archivo
GrabaHTK(char(Destino(r)),xest,N,sampPeriod,sampSize,sampKind);

%Grafico resultados
if grafica,
   figure

   subplot(3,1,1);
   pcolor(x);
   xlabel('x: vector limpio');
   title('Resultados');
   shading flat
   %colorbar;

   subplot(3,1,2);
   pcolor(y);
   xlabel('y: vector sucio');
   shading flat
   %colorbar;
```

```
        subplot(3,1,3);
        pcolor(xest);
        xlabel('xest: vector limpio estimado');
        shading flat
        %colorbar;
    end,
    %-------------------------------------------------------------------------------
    %Calculo algunos indicadores de desempeño

    %Calculo el ECM
    ECM=sqrt(sum(sum((xest-x).^2)))/(L*N)

    %Calculo SNR
    ruido=y-x;
    ruidonnlf=xest-x;
    %Energía del ruido original
    eruido=sum(sum(ruido.^2));
    %Energía del ruido del proceso de denoissing
    eruidonnnlf=sum(sum(ruidopof.^2));
    %Energía de la senial
    esenial=sum(sum(x.^2));

    snrpof=10*log10(esenial/eruidonnnlf)
    snroriginal=10*log10(esenial/eruido)

    %Calculo la medida de distorsion relativa
    d=sqrt(mean(ruido'.^2)./var(x'))

    toc
end
diary

%EOF NNNLFRUN.M
```

```
%Esta funcion carga los datos de origen (x) y destino (y) para el mapeo mediante POF y NNLF
function[x,y,N,L,sampKind]=CargaDatosxy(path)
if nargin < 1
   path='\\AZULYORO\lrufine\Mis_documentos\Denoising\Datos\';
   datosx='limpios\caract\mfcc.logE\'; %directorio datos origen
   datosy='sucios\snr10db\caract\mfcc.logE\'; %directorio datos destino
   speakers='af14_001*.mfc';
end;
%-------------------------------------------------------------------------
%Creo la lista de archivos origen a procesar
disp(['Leyendo datos origen']); %Muestro el hablante
lista=dir([path datosx speakers]);
%Leo la lista
x=[]; Nx=0;
for i=1:length(lista)
  speaker = lista(i).name;
  disp(['Hablante: ' speaker]); %Muestro el hablante
  %Llamo a LeeHTK con los parametros corresp.
  [xaux,Naux,sampPeriod,sampSize,sampKind]=LeeHTK([path datosx speaker]);
  %Acumulo los datos
  x=[x xaux];  Nx=Nx+Naux;
end
%-------------------------------------------------------------------------
%Creo la lista de archivos destino a procesar
disp(['Leyendo datos destino']); %Muestro el hablante
lista=dir([path datosy speakers]);
%Leo la lista
y=[]; Ny=0;
for i=1:length(lista)
  speaker = lista(i).name;
  disp(['Hablante: ' speaker]); %Muestro el hablante
  %Llamo a LeeHTK con los parametros corresp.
  [yaux,Naux,sampPeriod,sampSize,sampKind]=LeeHTK([path datosy speaker]);
  %Acumulo los datos
```

```
  Ny=Ny+Naux;
end;
  y=[y yaux];
%-------------------------------------------------------------------------
if Ny~=Nx,
   disp('Error : Cantidad de datos en x y y no concuerdan!');
else
   N=Nx;
end;

%La dimensión de los vectores de caracteristicas es el tamaño total de la muestra
%dividido el tamaño de cada valor (real de cuato bytes o float32)
L=sampSize/4;

%EOF CARGADATOSXY.M
```