Práctica 4: Reconocimiento y detección del hablante

Procesado de Audio y Voz, Antonio Bonafonte

Òscar Lorente
Ingeniería de Tecnologías y Servicios de Telecomunicación
Barcelona, España
oscar.lorente.co@gmail.com

Julio González
Ingeniería de Tecnologías y Servicios de Telecomunicación
Barcelona, España
jgonzal199842@gmail.com

***Abstract*—El objetivo de esta práctica es implementar un sistema de reconocimiento, clasificación y verificación del hablante, basado en los coeficientes cepstrales MFCC y el uso de los *Gaussian Mixture Models* (GMM). El programa -implementado en *C++*- consta básicamente de cuatro partes: extracción de características del locutor (LPC, LPCC o MFCC), implementación y uso de un algoritmo de estimación y evaluación del modelo probabilístico GMM a partir de las características extraídas, clasificación del hablante mediante el uso de un clasificador *Bayesiano* y por último verificación de locutor. El sistema de reconocimiento y verificación del hablante implementado presenta unos resultados excelentes, con un porcentaje de error del 0.13% (1 error sobre 785 casos) en el caso del clasificador y un coste de 1.6 respecto a la verificación del locutor, resultando ser un sistema muy robusto y de notables prestaciones. Finalmente, se ha desarrollado y puesto en práctica un algoritmo de Deep Learning basado en un perceptrón multicapa (red neuronal artificial multicapa) para clasificación del hablante, estudiando la influencia de distintos parámetros -función de activación, número de unidades por capa, *leraning rate*...- en la evaluación final de este, que ha resultado ser un sistema con muy buenas prestaciones (porcentaje de error del 0.25%: 3 errores sobre 785).**

I. Introducción

El sistema de reconocimiento y detección del hablante implementado durante esta práctica está basado en el análisis de la envolvente espectral de la señal de audio y voz. Tras extraer las características de las señales de voz necesarias (coeficientes LPCC o MFCC), el sistema modela la función de probabilidad de cada hablante (mediante los coeficientes extraídos) para realizar un clasificador Bayesiano mediante el modelo GMM (*Gaussian Mixture Model*). Este modelo será entrenado (creando un GMM para cada hablante) para finalmente clasificar a los locutores, y verificar si son legítimos o no. La verificación se realizará de forma binaria, permitiendo o denegando el acceso a las señales de entrada mediante el uso de un *score* para cada uno de los ficheros de la base de datos. Dicho *score* se obtendrá utilizando la probabilidad de que esta señal de entrada se corresponda con un locutor (estimada mediante los GMM) normalizada respecto a un modelo general (*world*), para finalmente verificar si el *score* calculado está por encima de un *threshold* (usuario legítimo) o no (impostor). Este *threshold* no deberá ser demasiado alto, ya que en ese caso los usuarios legítimos podrían tener problemas para acceder al sistema, ni demasiado bajo, pues los impostores podrían superar el sistema con facilidad. Evidentemente, y debido a la importancia y popularidad cada vez mayor que estos sistemas están obteniendo en los últimos años, la seguridad y robustez en estos es un factor importante, y siempre se priorizará que ningún locutor ilegítimo (impostor) supere el sistema de reconocimiento.

Adicionalmente, se ha realizado una mejora en la implementación del sistema de reconocimiento y clasificación del hablante basada en una Deep Neural Network. En los últimos años el Deep Learning ha estado en boca de gran parte de la comunidad científica y tecnológica, demostrando unos resultados y rendimiento sobresalientes y muy por encima de las tecnologías anteriores. Es por este motivo que hemos decidido introducirnos en este ámbito de la tecnología desarrollando un sistema clasificador del hablante (Speaker classifier) utilizando un perceptrón multicapa (red neuronal artificial multicapa). A la hora de desarrollar el sistema, se ha estudiado la influencia de ciertos parámetros como las diferentes funciones de activación (ReLU, LeakyReLU, ReLU6), número de unidades en capas ocultas (20, 50, 70, 100) y optimizadores (Adam, SGD, Adagard), de entre otros, para determinar qué modelo de red neuronal ofrece las mejores prestaciones en la aplicación presente.

La memoria de esta práctica consta de los siguientes apartados, en los que se explicará en mayor profundidad las metodologías utilizadas y resultados obtenidos:

- Descripción del diseño
  - Extracción de características
  - Estimación GMM y Reconocimiento del Hablante
  - Verificación del Locutor
  - Evaluación
- Resultados obtenidos
- Speaker Identifier with Deep Neural Networks
- Conclusiones

II. Descripción del diseño

A. Extracción de características

El primer paso a la hora de implementar el algoritmo de reconocimiento de locutor es diseñar un sistema de extracción de características para obtener los coeficientes más indicados para este tipo de aplicación. En este caso, y mediante el uso

de la librería SPTK (SPeech ToolKit), se han extraído los coeficientes LPC, LPCC y MFCC de un fichero de audio (Fig. 1) para determinar qué coeficientes son los más incorrelados entre ellos, y por lo tanto, más indicados para la clasificación.

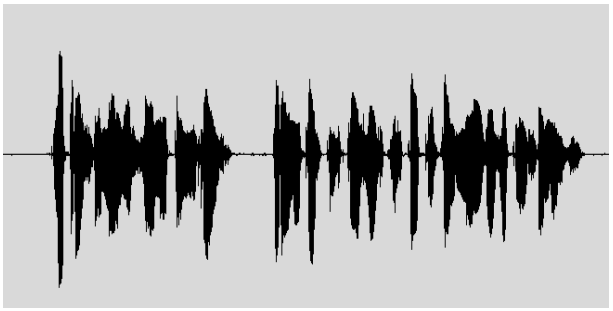

Fig. 1. Señal de audio utilizada para extraer y comparar los coeficientes LPC, LPCC y MFCC.

- ***LPC (Linear Prediction Coefficients)***: La predicción lineal LPC se basa en la predicción de las muestras de una señal (x[n]) mediante la combinación lineal de las muestras anteriores:

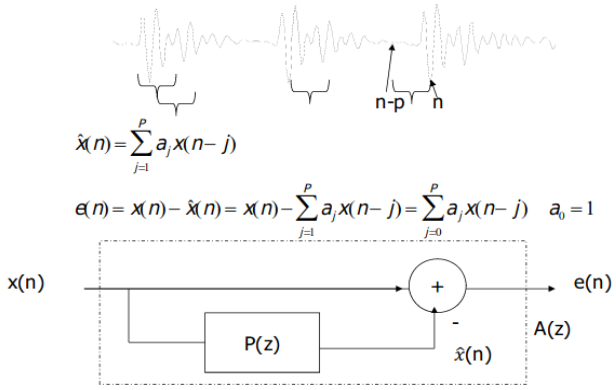

Fig. 2. Sistema de predicción lineal para extracción de coeficientes LPC.

Los coeficientes de la predicción lineal (Fig. 2) son estimados para minimizar la potencia del error.

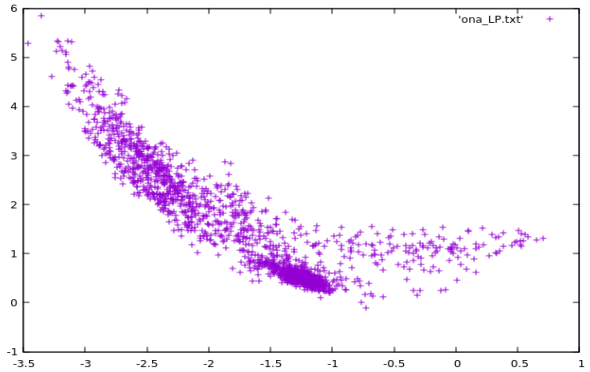

Fig. 3. Coeficientes LPC de orden 8 de la señal de audio.

Como puede apreciarse (Fig. 3) los coeficientes obtenidos mediante el LPC son altamente correlados, presentando una distribución casi lineal, por lo que no son adecuados para un clasificador del hablante.

- ***LPCC (LPC-based Cepstral Coefficients)***:
Los coeficientes LPCC -o coeficientes cepstrales basados en LPC- nos permiten obtener los coeficientes cepstrales de una señal de audio mediante filtros de predicción lineal, según las expresiones matemáticas de la Fig. 4.

$$c_o = \ln \sigma^2$$
$$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \quad 1 \le m \le p$$
$$c_m = \sum_{k=m-p}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \quad m > p$$

Fig. 4. Obtención de los coeficientes cepstrales mediante LPC.

Como se puede observar en la representación 2-D de los coeficientes LPCC (Fig. 5), estos son claramente más incorrelados que los LPC, resultando ser algo mas adecuados para la aplicación en cuestión..

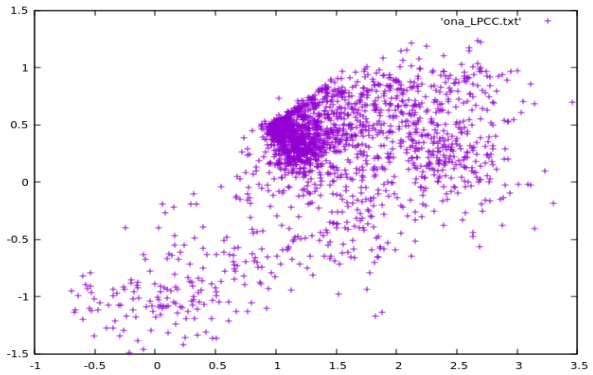

Fig. 5. 12 coeficientes LPCC de orden 8 de la señal de audio.

- ***MFCC (Mel Frequency Cepstral Coefficients)*** :
Los coeficientes MFCC se obtienen mediante el sistema representado en la Fig. 6, y se calculan utilizando un banco de filtros (Mel Filter Bank) que intenta aproximar el comportamiento de la percepción del oído humano (no lineal en frecuencia), obteniendo así unos coeficientes muy útiles en aplicaciones del procesado de audio y voz.

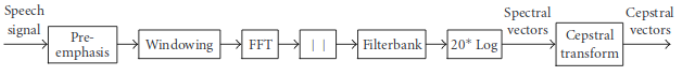

Fig. 6. Representación de la parametrización de los coeficiantes cepstrales basados en el banco de filtros.

En este caso, los coeficientes resultantes (Fig. 7) son altamente incorrelados entre sí y se observa una gran dispersión en su representación bidimensional, lo que

indica que los coeficientes MFCC son los óptimos para la aplicación del reconocimiento del hablante.

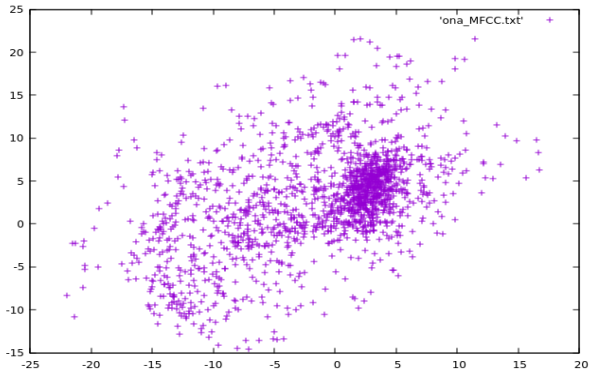

Fig. 7. 16 Coeficientes MFCC obtenidos utilizando un banco de 24 filtros.

En conclusión, y observando las representaciones 2-D de las características LPC, LPCC y MFCC adjuntas, se puede asegurar que los coeficientes más indicados para la aplicación del reconocimiento del hablante son los MFCC: Mel-Frequency Cepstral Coefficients, pues son los más incorrelados e incorporan un banco de filtros que simula el comportamiento de la percepción del oído humano.

B. Estimación Gaussian Mixture Models (GMM) y Reconocimiento del Hablante

En el sistema de reconocimiento del hablante, una vez seleccionadas las características más indicadas para la aplicación - coeficientes MFCC, y con el propósito de realizar un clasificador *Bayesiano* para decidir qué hablante se corresponde con qué fichero de audio, se debe modelar la función de probabilidad de estas características para cada locutor. El método escogido para modelar estas funciones de probabilidad son los *Gaussian Mixture Models* (GMM), que dados unos datos de entrenamiento (vector de características) $x = x_1, ..., x_n$, estima la probabilidad de que este vector de características se corresponda con un locutor θ como:

$$L(\theta) = p(X \mid \theta) = p(x_1, ..., x_n \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta) =$$

$$\prod_{i=1}^{n} \sum_{k=1}^{k} c_k p(x \mid m_k) = \prod_{i=1}^{n} \sum_{k=1}^{k} c_k N(x_i, \mu_k, \Sigma_k)$$

Es decir, definiendo esta probabilidad como una suma finita de distribuciones Gaussianas (de media y varianza distintas) y unos pesos asociados a cada una de ellas. El objetivo de las *Gaussian Mixture Models* es, tras haber inicializado θ (mediante uno de los tres métodos expuesto a continuación), iterar mediante el algoritmo de *Expectation-Maximization* para ir ajustando los pesos asociados a cada Gaussiana (c_k) y los parámetros de esta (μ_k y Σ_k) para maximizar *p(X‖θ)* (Maximum Likelihood Estimation).

El modelo GMM puede ser inicializado de tres modos:

- ***Random***: Inicialización más sencilla pero menos eficaz debido a que no busca características incorreladas entre sí, desaprovechando las ventajas que esto supone cuando se obtienen las características.

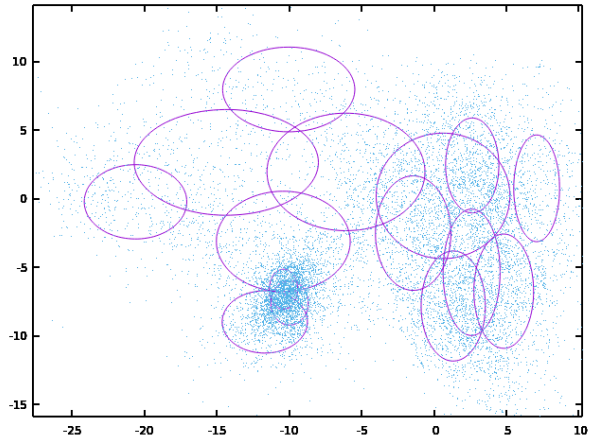

Fig. 8. Modelo GMM para un locutor mediante el uso de 15 Gaussianas e inicialización Random.

- ***EM-Split*** (*Expectation-Maximization*): Método iterativo, que tras un número elevado de iteraciones acaba convergiendo, utilizando pocas Gaussianas para ello. Los resultados suelen ser buenos - mejores que en el caso Random, pero se necesitan muchas iteraciones para llegar a ellos.

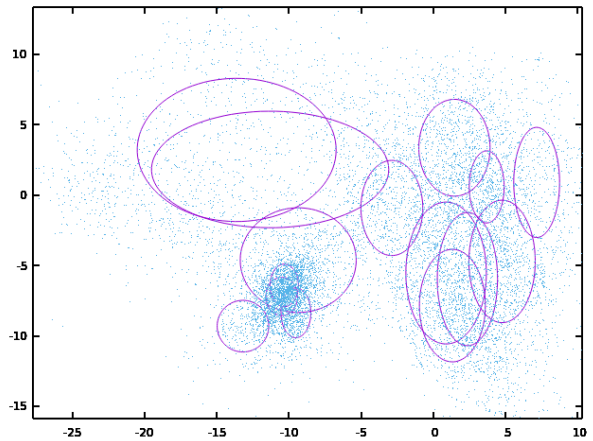

Fig. 9. Modelo GMM para un locutor mediante el uso de 15 Gaussianas e inicialización EM_split.

- ***VQ*** (*Vector Quantifier*): Método que normalmente se usa para la inicialización de los parámetros en las GMM, pues no necesita muchas iteraciones ni demasiadas gaussianas, y permite mayor rapidez y optimización del sistema que los anteriores, ofreciendo mejores resultados.

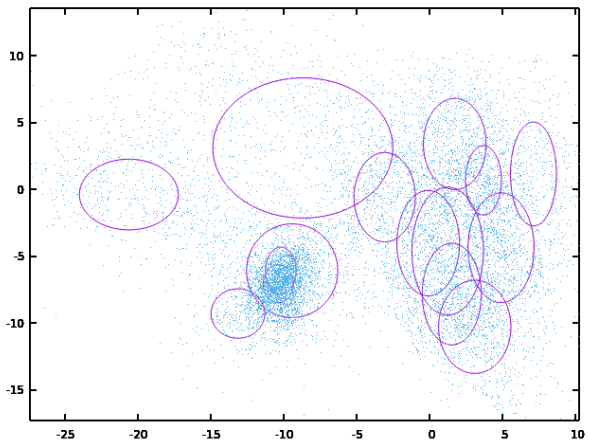

Fig. 10. Modelo GMM para un locutor mediante el uso de 15 Gaussianas e inicialización VQ.

El método escogido tras estudiar los resultados y tiempo necesario para la inicialización en cada uno de los casos ha sido el *VQ*, que con una convergencia rápida y computacionalmente ligera presenta las mejores prestaciones. Llegados a este punto, y una vez finalizado de forma satisfactoria el modelo *GMM* que estima las probabilidades de los vectores de características del sistema respecto a cada locutor mediante el algoritmo EM, ya se puede utilizar dicho sistema como clasificador del hablante y estudiar la influencia de distintos parámetros en los resultados finales, tal y como se profundiza en el apartado de *Evaluación*.

C. Verificación del Locutor

Por otro lado, se ha implementado un programa de verificación del locutor que analiza la voz a su entrada y la información sobre el usuario para decidir si el usuario correspondiente a la señal de entrada es legitimo o un impostor. Para llevar a cabo esta decisión, se calcula un score *s* para cada fichero de audio de test y se compara con un *threshold* predefinido *th* para determinar si el acceso al usuario se le es concebido o denegado:

$$s < th \rightarrow ACCESS\_DENIED$$

$$s \geq th \rightarrow ACCESS\_GRANTED$$

Así pues, los posibles errores que el sistema puede propiciar son: fallo de detección (*miss*), se le ha denegado el acceso a un usuario legítimo, y falsa alarma, se le ha autorizado el acceso a un impostor. Teniendo en cuenta que una de las aplicaciones típicas de este sistema es la seguridad en controles de acceso, lo más conveniente es definir un valor de *threshold* alto para minimizar las falsas alarmas (impostores) y darle menos importancia a los fallos de detección (que no presentan *a priori* ningún peligro). Para modelar el *coste* del sistema en función de los errores cometidos, se define la siguiente expresión:

$$C = \frac{1}{min(\rho, 1-\rho)}(\rho \cdot p_m + (1-\rho) \cdot p_{fa}) \cdot 100$$

donde:

- p_m: porcentaje de fallos del sistema (bloqueo) frente a usuarios legítimos
- p_{fa}: porcentaje de fallos del sistema (acceso) frente a impostores
- ρ: parámetro de definición del objetivo (target). En este caso ρ = 0.01 para penalizar mucho los accesos sin autorización (una flasa alarma cuenta como 99 bloqueos a usuarios legítimos).

El *coste*, tal y como está definido, es el parámetro que se usa como referencia a la hora de evaluar el sistema de verificación del locutor, siendo el objetivo principal el de obtener un coste lo más bajo posible (haciendo énfasis en reducir a 0 el número de falsas alarmas) para garantizar un sistema de calidad y robusto frente a impostores.

Por otra parte, para modelar el *score* se utiliza la probabilidad del GMM del usuario normalizada respecto un modelo general (*mundo* o *background*):

$$s = \frac{f_{\theta_u}(x)}{f_{\theta_w}(x)}$$

donde:

- *s*: *score*, magnitud utilizada para decidir si se permite el acceso
- $f_{\theta u}$(x): modelo mediante GMM de las características acústicas del usuario
- $f_{\theta w}$(x): modelo mediante GMM de las características acústicas del conjunto de usuarios, *world*

D. Evaluación

El ultimo paso para la implementación correcta del sistema de reconocimiento y detección del hablante es el de la evaluación de los resultados obtenidos, que reflejaran la eficacia y robustez frente a impostores de nuestro sistema. Así pues, el objetivo principal es tanto el de minimizar el porcentaje de error en la clasificación como el de reducir lo máximo posible el *coste*, anteriormente definido, de nuestro sistema. Para llevar a cabo la evaluación de nuestro sistema, primeramente se entrenan los GMM mediante una base de datos de entrenamiento, que utilizando el algoritmo iterativo *Expectation-Maximization* ajusta los parámetros correspondientes (peso de cada Gaussiana y sus medias y varianzas) de cada uno de ellos para modelar mejor las características respecto a los locutores. A continuación se utilizan bases de datos de validación para comprobar que los pesos anteriormente computados son adecuados, y finalmente se hace uso de diversos ficheros de test, tanto de usuarios legítimos como de usuarios impostores, y se comprueba que el sistema entrenado sea generalizable: que no solo ofrezca buenas prestaciones para las señales utilizadas durante el entrenamiento, sino que

para cualquier tipo de señal de entrada el sistema puede funcionar adecuadamente. El resultado de todo este proceso de evaluación puede verse reflejado en el porcentaje de error en la aplicación de reconocimiento del locutor (comparando los aciertos del sistema con el total de señales de entrada, 785):

```
nerr=1  ntot=785        error_rate=0.13%
```

Fig. 11. Evaluación de la etapa de clasificación del sistema

y en el *coste* total del sistema verificador de locutor:

```
================================================
THR: 0.750461310675302
Missed:     4/250=0.0160
FalseAlarm: 0/1000=0.0000
------------------------------------------------
==> CostDetection: 1.6
================================================
```

Fig. 12. Evaluación del sistema verificador de locutor

Ambos resultados son comentados y profundamente desarrollados en el siguiente apartado: Resultados Obtenidos.

III. Resultados obtenidos

El objetivo principal del proyecto desarrollado es diseñar e implementar un sistema de reconocimiento y verificación del hablante totalmente generalizado para que, independientemente del audio y/o locutor que se analice se obtengan unos porcentajes de error y coste mínimos. Consecuentemente, y con el propósito de minimizar dichos errores, se han realizado un gran número de pruebas distintas durante el desarrollo del sistema propuesto, ajustando distintos parámetros (número de coeficientes MFCC, número de filtros del *Mel Filter Bank*, número de Gaussianas utilizadas en el GMM, etcétera) hasta conseguir reducir el porcentaje de error al 0.13% (1 fallo sobre 785 casos) y el coste a 1.6 (con 4 *misses* y 0 falsas alarmas). A continuación, para exponer el camino que hemos seguido hasta llegar a los resultados obtenidos, se presentan las distintas pruebas que hemos realizado durante el diseño y desarrollo del sistema. Primeramente, aunque ya habíamos seleccionado los coeficientes MFCC como los más indicados para esta aplicación debido a su incorrelación, queríamos comprobar experimentalmente si, utilizando el mismo número de Gaussianas (*mixtures*) para el GMM, los coeficientes MFCC presentaban mejores prestaciones que los coeficientes LPCC.

TABLE I

Método	Nº coef.	Nº filtros / orden LPC	Nº Mixt.	Error (%)
LPCC	15	10	5	4.08
LPCC	15	10	8	2.04
MFCC	12	20	5	1.02
MFCC	12	20	8	0.38

Efectivamente, empleando los coeficientes MFCC, independientemente del número de Gaussianas utilizadas, se obtiene un porcentaje de error más bajo que en el caso de los coeficientes LPCC (Table I).

Una vez escogido el tipo de coeficientes a utilizar, MFCC, se ha optado por, manteniendo el número de coeficientes (12) y de filtros (20), experimentar con distintos parámetros para minimizar el error. A raíz de esto (Table II), se ha obtenido que utilizando 8 Gaussianas y 20, 15 o 10 iteraciones en el algoritmo de Exceptation-Maximization (EM) se ha reducido el porcentaje de error al 0.38% (3 errores sobre 785).

TABLE II

MFCC / 12 coef. / 20 filtros		
Nº Mitxtures	Nº Iteraciones	Error (%)
5	20 / 15	1.02
5	10	1.40
8	20 / 15 / 10	0.38
8	5	0.89

Llegados a este punto, y con unos resultados de reconocimiento del hablante muy satisfactorios, decidimos centrarnos en estudiar la influencia de estos parámetros en el coste, fallos de detección y falsas alarmas del sistema verificador de locutor. Por ese motivo, con 12 coeficientes MFCC, 20 filtros y 8 Mixtures, se comprueba la influencia del número de iteraciones del algoritmo EM en el coste (*C*), los fallos de detección (*Miss*) y las falsas alarmas (*FA*).

TABLE III

MFCC / 12 coef. / 20 Filtros / 8 Mixtures				
Nº Iter.	Error (%)	Miss	FA	Coste
20	0.38	9	1	13.5
15	0.38	16	1	16.3
10	0.38	24	1	19.5

Es curioso observar como, aún manteniendo el porcentaje de error al 0.38 como sistema de reconocimiento del habla, el hecho de variar el número de iteraciones entre 20, 15 y 10 afecta a los resultados obtenidos en el verificador de locutor, siendo el caso de 20 iteraciones el que ofrece mejores prestaciones (Table III). Aún así, teniendo en cuenta que uno de los principales objetivos era reducir el número de falsas alarmas (*FA*) a 0, se ha seguido experimentando con ciertos parámetros, esta vez manteniendo el uso de los coeficientes MFCC y 20 iteraciones de EM, y haciendo énfasis en la influencia del número de coeficientes MFCC y el número de filtros del *Mel Filter Bank*.

TABLE IV

MFCC / 20 Iteraciones						
Nº coef.	**Nº Filtros**	**Nº Mix.**	**Error(%)**	**Miss**	**FA**	**Coste**
12	20	8	0.38	9	1	13.5
16	20	8	0.38	19	0	7.6
16	24	8	0.25	23	0	9.2
16	24	15	0.13	4	0	1.6

Finalmente, tras haber reducido el porcentaje de errores del clasificador del hablante a un 0.13% (1 fallo sobre 785), y haber minimizado el coste del verificador del locutor hasta un valor de C = 1.6, con 4 *misses* y 0 falsas alarmas (Table IV), concluimos que el sistema diseñado e implementado ofrece unas excelentes prestaciones con los parámetros concretos: 16 coeficientes MFCC, 24 filtros, 15 Gaussianas y 20 iteraciones del algoritmo *EM*.

IV. Speaker Identifier with Deep Neural Networks

Finalmente, y como apliación del sistema diseñado, se ha implementado un clasificador del hablante mediante el uso de *Deep Neural Networks*, es decir, redes neuronales de Deep Learning. Concretamente, el algoritmo desarrollado se basa en un perceptrón multicapa (Fig. 13) que recibe como *Inputs* los ficheros *.mcp* que contienen los modelos GMM (que han sido entrenados a partir de los coeficientes MFCC previamente calculados y utilizando el algoritmo de Expectation Maximization) y ofrece como *Output* el locutor correspondiente.

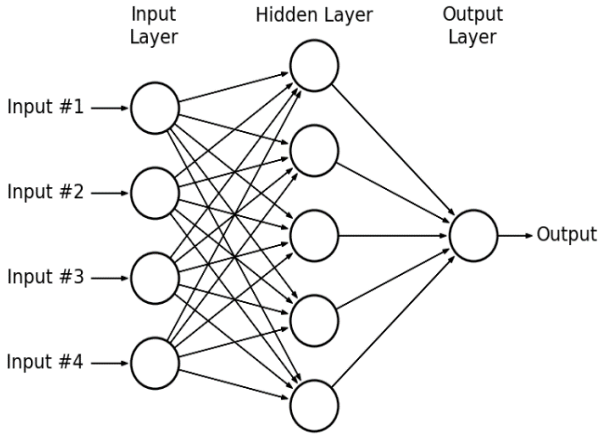

Fig. 13. Ejemplo de perceptrón multicapa (falta el Bias y los pesos a la salida de cada neurona).

Las capas de la red neuronal aplican una transformación lineal (son de tipo *Linear*) a los datos de entrada utilizando unos pesos w y bias b para cada neurona. Por otro lado, la función de activación no-lineal utilizada en la *Output Layer* es una *Softmax*, mientras que en las capas ocultas (que en este caso hay 2) se utilizan funciones como la *ReLU* o *LeakyReLU*. En cuanto al aprendizaje, el sistema implementado utiliza el método de *backpropagation* para, una vez obtenido el locutor en la *Output Layer* y comprobado si este locutor se corresponde o no con el correcto (aprendizaje supervisado), ajustar los pesos w de las distintas capas ocultas siguiendo distintos criterio de optimización: Adam, SGD o Adagard.

Se han realizado distintas pruebas para comprobar la influencia de distintos parámetros de la capa (número de unidades por capa, función de activación...) en los resultados ofrecidos por la red neuronal implementada. Para empezar, y modificando el número de neuronas por capa junto con el *Learning Rate*, se puede comprobar (Table V) como para el caso de 100 neuronas por capa y un ratio de aprendizaje de 0.001, se obtiene un 0.25% de porcentaje de error (3 errores sobre 785), ofreciendo muy buenas prestaciones. Es lógico pensar que aumentar el número de unidades por capa aumenta el coste computacional (número de operaciones, tiempo de ejecución), pero los resultados obtenidos son más que satisfactorios.

TABLE V

Deep Learning / ReLU / Optim.Adam			
Nº Unidades /capa oculta	**Learning Rate**	**Nº Correctas (Total: 785)**	**Error(%)**
20	0.001	749	4.59
50	0.001	780	0.64
70	0.001	782	0.38
100	0.001	783	0.25
100	0.01	777	1.02
100	0.0001	781	0.51

Por otro lado, utilizando un learning ratio de 0.001 y 100 unidades por capa, se ha comprobado (Table VI) como las funciones de activación ReLU y LeakyReLU ofrecen un porcentaje de error mínimo (0.25%) respecto otras funciones como ReLU6, en cuyo caso el error se ve aumentado sustancialmente.

TABLE VI

Deep Learning / LR: 0.001 / Optim.Adam / 100 Unidades por capa oculta		
Función de Activación	**Nº Correctas (Total: 785)**	**Error(%)**
ReLU	783	0.25
LeakyReLU	783	0.25
ReLU6	690	12.10

Por último, manteniendo el learning ratio en 0.001, 100 unidades por capa y ReLU como función de activación, se ha querido comprobar (Table VII) la influencia del optimizador utilizado a la hora de ajustar los pesos durante el *backpropagation*, comparando los optimizadores SGD (Stochastic Gradient Descent), Adam y Adagrad, siendo los dos últimos los que mejores resultados ofrecen, aunque estando el optimizador Adam muy por encima respecto al Adagrad (0.25% respecto 4.46%).

Finalmente, en la imagen adjunta se representa la evolución del *loss*

TABLE VII

Deep Learning / LR: 0.001 / ReLU / 100 Unidades por capa oculta		
Optimizador	**Nº Correctas (Total: 785)**	**Error(%)**
Adam	783	0.25
SGD	694	11.59
Adagrad	750	4.46

V. Conclusiones

En el desarrollo de ésta práctica hemos podido experimentar de primera mano la complejidad y metodología de implementación de un sistema de reconocimiento y detección del habla, y su eficacia y posibles aplicaciones. Para la implementación del sistema, se ha experimentado con diversos métodos de extracción de características de una señal de voz (LP, LPCC y MFCC), se ha estudiado el comportamiento del sistema en funcións tres tipos de inicialización de un modelo GMM (*Random*, *EM-Split* y *VQ*) y la influencia de distintos parámetros (número de coeficientes, filtros, mixtures e iteraciones) en los resultados obtenidos. Por otro lado, también se ha comprobado la importancia de la definición del *threshold* i el cálculo del *coste* a la hora de verificar el locutor de la señal de entrada. Adicionalmente, hemos sido introducidos a las nuevas tecnologías de Deep Learning, incorporando una mejora basada en un perceptrón multicapa, y experimentando con los diferentes parámetros que creemos más determinantes. Los resultados han podido ser analizados mediante el uso de *scripts .sh* (intérprete Bash) para determinar la eficacia y calidad del sistema, resultando en ambos casos en excelentes prestaciones. Así pues, asumiendo las múltiples opciones que el uso de modelos GMM y tecnología Deep Learning aportan al procesado de audio y voz, las posibilidades a la hora de implementar nuevas aplicaciones y funcionalidades prácticas siguen aumentando y, cada vez más, podemos comprobar el enorme potencial de estos sistemas.

References

[1] A Tutorial on Text-Independent Speaker Verification. (n.d). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.362.3704&rep=rep1&type=pdf.

[2] Comparative Analysis of LPCC, MFCC and BFCC for the Recognition of Hindi Words using Artificial Neural Networks. (n.d). Retrieved from https://pdfs.semanticscholar.org/a9d5/3dce0ef368d9bb0e461ad73a4519319e79a6.pdf.

[3] A Comparative Study Of LPCC And MFCC Features For The Recognition Of Assamese Phonemes. (n.d). Retrieved from https://pdfs.semanticscholar.org/7c8f/8a9d5ba85788b569bc04ca9f07d6ce689e8c.pdf.