Universidad Zaragoza
1542

ESTUDIO DE INTEGRACIÓN DE CLASIFICADORES DE RASGOS FONÉTICOS PARA LA MEJORA DE SISTEMAS DE RECONOCIMIENTO DE GRAN VOCABULARIO

PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIÓN

ESPECIALIDAD: COMUNICACIONES

JULIO 2011

AUTOR: CAROLINA VILLALTA PÉREZ

DIRECTOR: DR. ANTONIO MIGUEL ARTIAGA

instituto de investigación en ingeniería de Aragón

CPS

1542 Departamento de Ingeniería Electrónica y Comunicaciones
Universidad Zaragoza

RESUMEN

En los sistemas de inteligencia ambiental una parte fundamental la constituye el interfaz hombre-máquina, y dentro de éste, la interacción oral en ambos sentidos, de la que forman parte los sistemas tanto de reconocimiento automático como de síntesis de voz. En sistemas complejos la interacción simple por medio de comandos limita las posibilidades de un sistema de inteligencia ambiental, por ello es preciso tener disponibles reconocedores de voz de gran vocabulario.

El trabajo de investigación propuesto tiene como objetivo la mejora de las prestaciones de un reconocedor automático de voz de gran vocabulario, medidas en tasa de errores de palabra. Para ello la investigación atenderá principalmente a la mejora del modelo acústico, dejando el de lenguaje por defecto. Las propuestas que se plantean consistirán en realizar clasificadores de rasgos fonéticos, en una fase previa a la de reconocimiento, y concatenar las salidas de estos clasificadores con los vectores de características utilizados comúnmente para facilitar el reconocimiento. Se propondrán diversas maneras de utilizar estos nuevos vectores de características como entradas al reconocedor.

Los clasificadores estarán especializados en la distinción de ciertos fonemas o grupos de fonemas, e incluso aspectos articulatorios o contextuales de la producción de la voz. En concreto desarrollaremos siete tipos de clasificadores. Se propone realizar los clasificadores con redes neuronales artificiales o máquinas de soporte vectorial, a través de la experimentación con la base de datos TIMIT y distintos tipos de parametrización de la señal de voz, se elegirá el tipo de clasificador a utilizar que tendrá que cumplir unos requisitos. Una vez elegido los clasificadores se utilizarán para clasificar la base de datos WSJ0. Utilizaremos para el reconocimiento el reconocedor HTK y una modificación de este, el reconocedor HTK_STUZ desarrollado por el grupo GTC del I3A. Las pruebas consistirán en introducir directamente el vector de características formado por las salidas de los clasificadores junto con los vectores comúnmente utilizados (características tandem); después se adaptarán las salidas de los clasificadores a los modelos estadísticos utilizados en HTK y por último, y como novedad en el estado del arte de los reconocedores de gran vocabulario se utilizarán nuevos modelos estadísticos en el reconocimiento de estas características tandem, junto con los utilizados por el HTK. Los modelos que se usarán son: distribución Beta, redes Bayesianas y distribución de Bernoulli.

AGRADECIMIENTOS

Este proyecto ha sido apoyado por el Instituto de Investigación en Ingeniería de Aragón (I3A), a través de la beca de iniciación a la investigación durante el periodo comprendido entre Octubre del 2010 y Abril del 2011.

Primero quisiera agradecer el apoyo y la ayuda recibida durante todo el tiempo que ha durado este proyecto a mi director Antonio Miguel Artiaga. También me gustaría darles las gracias a una serie de personas que me han apoyado no solo durante el proyecto sino también durante toda la carrera. Gracias a Gonzalo, mis padres Joaquín y Nuria, mi hermana Nuria, mis abuelos Antonio y María, y por último a todos mis amigos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Introducción

1.1 Objetivo del proyecto

En los sistemas de inteligencia ambiental una de las partes fundamentales es el interfaz hombre-máquina, y dentro de éste, la interacción oral en ambos sentidos, de la que forman parte los sistemas de síntesis y reconocimiento automático de voz. Para hacer posible un sistema de inteligencia ambiental, la interacción simple por medio de comandos no es suficiente, por ello es necesario disponer de reconocedores de voz de gran vocabulario en estos sistemas que permitan una interacción oral entre hombre y máquina lo más parecida posible a una conversación entre dos humanos. Pero no solo se tiene como objetivo los entornos de inteligencia ambiental, sino también dotar de accesibilidad a usuarios con dificultades en el manejo de ordenadores, discapacitados o profesionales que buscan documentación dentro de toda la red de información, documentación y bases de datos multimedia, en especial los de relevancia cultural e informativa. El acceso a todo este patrimonio cultural, histórico, social e informativo que se encuentra en forma de grabaciones recopiladas a lo largo de los últimos años es una labor larga y tediosa, y muchas veces el acceso a un fragmento específico de estas grabaciones se hace prácticamente imposible. Dentro de este contexto las tecnologías de la voz, y en concreto, el reconocimiento automático del habla de gran vocabulario está teniendo una gran relevancia.

Los Modelos Ocultos de Markov (HMM) [1] son la técnica más empleada en reconocimiento automático del habla (RAH). Aún hoy en día están lejos de alcanzar altas prestaciones en estos sistemas, por ello se han estudiado otras posibles técnicas o la unión de varias técnicas para mejorar estos resultados. El objetivo es utilizar clasificadores que sean más precisos y discriminativos, como por ejemplo las redes neuronales artificiales. A finales de los 80 y principios de los 90 se comienzan a usar las redes en RAH, primero como alternativa a los HMM y después como ayuda a estos, formando los sistemas Tandem [2]. También se han utilizado otro tipo de clasificadores como las máquinas de soporte vectorial (SVM) [3], y actualmente se están investigando las características Bottle-Neck [4]. Todo esto será explicado con más detalle en el Capítulo 2.

El proyecto tiene como objetivo la mejora de las prestaciones del reconocedor automático de voz de gran vocabulario, medidas en tasa de errores de palabras (WER). La investigación se centrará en mejorar el modelo acústico, dejando por defecto el modelo de lenguaje. Lo que se propone es utilizar clasificadores de rasgos fonéticos en una fase previa a la extracción de características, de forma que la salida de estos clasificadores se pueda concatenar a los vectores de características utilizados normalmente y obtener unos nuevos vectores de características que faciliten el reconocimiento. Estos clasificadores estarán especializados en la distinción de fonemas o grupos de fonemas, e incluso aspectos articulatorios o contextuales de la producción de la voz. A través de la experimentación con distintas bases de datos

elegiremos aquella opción u opciones que nos den mejor resultado. Para la extracción de características también probaremos con distintas opciones: los coeficientes MFCC (Mel Frecuency Cepstral Coefficients) [1], los coeficientes PLP (Predicción Lineal Perceptual) [5] o los coeficientes RASTA (Relative Spectral Transform) [6]. En el diseño de los clasificadores se ensayarán distintas alternativas como redes neuronales, mezclas de Gaussianas o máquinas de soporte vectorial (SVM). Respecto a los reconocedores utilizaremos el reconocedor HTK y el reconocedor del grupo AMI-GTC vivolab. En su integración se ensayarán también distintas alternativas como el uso de distribuciones Gaussianas, modelos gráficos o distribuciones Beta.

1.2 Revisión bibliográfica

El reconocimiento automático del habla (RAH) sigue siendo una tarea difícil a pesar de los avances de las últimas décadas, algunas de estas dificultades son: aumento del número de palabras en el diccionario, reconocimiento de voz continua, robustez frente a ruido, múltiples distorsiones, micrófonos, reconocedores independientes del hablante, etc. En particular los sistemas basados en modelos ocultos de Markov (HMM) proporcionan una manera simple y efectiva de modelar secuencias de vectores de características espectrales o temporales, por lo tanto son sin duda la técnica más popular usada en sistemas de reconocimiento automático de voz de gran vocabulario.

Un HMM es un modelo doblemente estocástico, que podemos representarlo como una secuencia de estados, donde cada estado tiene asociada una distribución de probabilidad, comúnmente la distribución elegida es la mezcla de Gaussianas. El objetivo del reconocedor es obtener la secuencia de estados más probable, dadas unas observaciones. Tienen alta tasa de reconocimiento en muchas circunstancias, como en: dictado, acceso a bases de datos, interfaz máquina-humano, acceso remoto a servicios a través de la línea telefónica y control de máquinas, pero siguen teniendo limitaciones importantes que no nos permiten aplicar RAH en algunas aplicaciones del mundo real, donde esta tasa se ve fuertemente degradada.

Durante finales de los 80 y los 90 se investigaron y probaron algunas alternativas a los HMM, la mayoría basadas en redes neuronales artificiales (ANN) [7,8], con el fin de superar las limitaciones de los HMM. Las ANN son unas técnicas discriminativas de clasificación, que poseen ciertas propiedades que las hacen muy adecuadas para utilizarlas en los problemas de clasificación de patrones en RAH, estas propiedades son principalmente: utilizan un criterio discriminativo en el aprendizaje; son aproximadores universales, esto es que son capaces de conseguir aproximaciones de funciones continuas con estructuras simples; y no requieren asumir las propiedades estadísticas de los datos de entrada así como la forma de la función de densidad de las salidas.

A pesar de obtener un buen funcionamiento en problemas de clasificación estáticos, presentan notables limitaciones al tratar con la clasificación de señales de voz. Con el fin de poder utilizar las redes en el problema de clasificación de señales de voz, los primeros sistemas perseguían la

adaptación de la arquitectura de la red neuronal a la estructura temporal de la voz. En este contexto, se propusieron dos clases de redes neuronales: Time-Delay Neural Networks (TDNN) [9] y Recurrent Neural Networks (RNN) [10]. Aunque estos sistemas han alcanzado buenos resultados en el reconocimiento de fonemas y palabras aisladas, las ANN no han obtenido éxito en tareas más complejas como en el reconocimiento de voz continua. La razón principal por la cual no han obtenido este éxito es por su incapacidad para modelar la variabilidad del tiempo de las señales de voz de forma tan eficiente como los HMM, que permiten además construir fácilmente sistemas de miles de palabras.

Para superar las dificultades comentadas en el párrafo anterior, muchos investigadores han propuesto los llamados sistemas híbridos ANN-HMM [11]. La idea básica es combinar en un solo sistema los HMM y las ANN para conseguir beneficiarnos de las propiedades de ambos sistemas: la habilidad de los HMM de modelar la variabilidad temporal de la señal de voz y la habilidad discriminativa de las ANN. Siguiendo este patrón se han desarrollado diferentes sistemas, de los cuales destacan los sistemas Tandem [13], que consisten en entrenar las redes neuronales para estimar la probabilidad a posteriori de los fonemas, y estas probabilidades utilizarlas como vectores de características para el reconocedor estándar GMM-HMM, alcanzando así buenos resultados en sistemas de contexto independiente. Numerosos estudios demuestran que estos sistemas híbridos alcanzan unos resultados equivalentes o incluso mejores en algunas tareas que los sistemas HMM [12,13]. También presentan un mejor comportamiento cuando la cantidad de datos de entrenamiento no es muy grande [13].

En esta última década ha aparecido una nueva herramienta en el campo del aprendizaje automático (Machine Learning), las máquinas de soporte vectorial (SVM) que pueden tratar con difíciles problemas de clasificación de patrones en varios campos de aplicación. Los SVM son clasificadores discriminativos con varias características a destacar: su solución es aquella que tiene margen máximo, es decir, maximizan la separabilidad de los datos; son capaces de tratar con datos de muy alta dimensión; y garantizan la convergencia al mínimo de la función de coste. Estas características de discriminación y generalización han hecho que los SVM sean populares y hayan tenido éxito.

Una de las razones por las que se propone su uso para RAH, es por su gran capacidad de generalización, que tendría que mejorar la robustez de los sistemas de RAH. Las redes neuronales y algunas modificaciones de los HMM minimizan el riesgo estructural, los SVM también lo hacen, pero también maximizan la distancia entre muestras y los límites de clasificación, por lo que tienen una mejor capacidad de generalización. La distancia máxima, conocida como el margen, es lo que hace que los SVM consigan una buena generalización, la solución de margen máximo les permite superar a la mayoría de clasificadores no lineales en presencia de ruido, el cual es una de los viejos problemas que tienen los sistemas de RAH [14]. A pesar de todas estas razones, las máquinas de soporte vectorial no son fáciles de usar para reconocimiento automático de voz. Tres son los principales problemas a superar: 1) SVM son originalmente clasificadores estáticos y por la tanto tienen que ser adaptados para tratar con las señales de voz, que son de duración variable; 2) Al principio fueron formulados como clasificadores binarios, mientras que en RAH tenemos problemas multiclase; y 3) Los algoritmos de entrenamiento actuales de los SVM no pueden manejar grandes bases de datos típicamente usadas en RAH a pesar de nuevas técnicas como Sparse SVM, el número de datos

de entrenamiento está todavía limitado a unos pocos miles. En [3] explica posibles soluciones para resolver los problemas planteados. Los SVM consiguen mejores prestaciones que los HMM en tareas con pequeñas bases de datos en condiciones de ruido, pero en tareas con grandes bases de datos aunque alcanza las mismas prestaciones que los HMM tienen un gran coste computacional de entrenamiento, debido sobre todo por la gran cantidad de datos que se utilizan en RAH.

Durante estos últimos años las características Tandem se utilizan como entradas al sistema GMM-HMM, llegándose a convertirse en una parte fundamental del estado del arte de los sistemas de reconocimiento automático de voz de gran vocabulario. Estas características son las salidas de la red neuronal que antes de introducirlas al sistema GMM-HMM son adaptadas a la estadística de este sistema utilizando la de-correlación PCA. En los últimos años se está investigando el uso de las características bottle-neck. Para obtenerlas se utiliza una red neuronal (Multilayer Perceptron,MLP) de 5 capas, donde la capa intermedia se llama bottle-neck. Una vez entrenada la red, lo que se toma como salida de esta son los valores obtenidos de la función de activación de la capa bottle-neck, y estas salidas son las características que introducimos al sistema GMM-HMM directamente. Sobre este tema y mejoras propuestas encontramos más información en [4].

1.3 Metodología de trabajo

En la realización del presente proyecto se han llevado a cabo distintas fases de trabajo, que incluyen una fase previa de documentación, una fase de aprendizaje de utilización de las distintas herramientas de clasificación, experimentación con los distintos clasificadores aprendidos y por último un análisis de los resultados obtenidos al aplicar estos clasificadores al reconocedor. A continuación se realizará una explicación con más profundidad de cada una de las fases.

1.3.1 Documentación

Esta fase previa se llevó a cabo durante los tres primeros meses del proyecto. El objetivo principal de esta fase consistía en conocer el estado del arte de los sistemas RAH, tanto comprender en qué consisten, sus distintas fases; así como conocer los distintos experimentos que se habían realizado; y sus futuras líneas de investigación.

1.3.2 Aprendizaje del uso de los clasificadores

Una de las fases más importantes del proyecto junto con la fase de experimentación. Ha consistido en aprender a utilizar los distintos tipos de clasificadores: mezclas de Gaussianas (GMM), máquinas de soporte vectorial (SVM) y redes neuronales (ANN).

El objetivo es utilizar las redes neuronales como clasificadores de rasgos fonéticos, pero para asegurarnos que nuestra elección es la correcta tenemos que comprobar los resultados que

obtenemos de las redes neuronales con los de las mezclas de Gaussianas, clasificadores utilizados comúnmente en tecnología de la voz, y con los SVM por sus posibles ventajas frente a las redes neuronales.

En el aprendizaje y desarrollo de las mezclas de Gaussianas la herramienta utilizada ha sido Matlab. Para comprender su funcionamiento y cuáles son sus parámetros clave para la obtención de un buen resultado se ha utilizado un ejemplo sencillo de clasificación, como es el caso de la XOR. Se ha partido de un código ya desarrollado.

En el caso de los SVM se ha utilizado la toolkit WEKA con la librería LibSVM [20], desarrollada por la universidad de Nueva Zelanda. En WEKA los SVM están implementados, así como otros tipos de clasificadores, y su utilización consiste en introducir los datos a clasificar y en darle los valores correctos a los parámetros del clasificador. La toolkit viene con un conjunto de datos-ejemplo muy útiles para aprender a manejar los clasificadores.

Por último para el aprendizaje de las redes neuronales se ha utilizado la mayor parte del tiempo de esta fase. Para entrenar las redes hemos utilizado la toolbox de redes neuronales de Matlab. Se comenzó también empleando el ejemplo de clasificación XOR, por su sencillez. Una descripción más detallada del uso de los SVM y las redes neuronales se da en el Capítulo 3.

1.3.3 Experimentos

Una vez que se aprendió a utilizar los distintos clasificadores con ejemplos sencillos se pasó a realizar pruebas un poco más complicadas. Estas pruebas solo se realizaron con los SVM y las redes.

Utilizamos la base de datos TIMIT [21], puesto que es una base de datos bastante grande. Las primeras pruebas se realizaron solo con una pequeña parte del conjunto de entrenamiento y de test de TIMIT, y consistían en clasificar algunos rasgos fonéticos. En esta fase ya nos pudimos dar cuenta de las limitaciones de los SVM al trabajar con grandes cantidades de datos.

El desarrollo de esta fase y también la de aprendizaje representan más del 50 % del trabajo realizado en el proyecto, aproximadamente 7 meses. En el Capítulo 3 se explicará con más detalle los experimentos realizados.

1.3.4 Análisis de resultados

La última fase consiste en el análisis de los resultados obtenidos al introducir nuestros clasificadores de rasgos fonéticos en el reconocedor. Se utilizó el reconocedor HTK [22] y el reconocedor vivolab, realizado por el grupo de tecnologías de la comunicación del I3A, para realizar las distintas pruebas. En esta fase la base de datos utilizada es la Wall Street Journal (WSJ0) [23], y su tasa de error de palabras obtenida en un sistema de reconocimiento estándar, va a ser usada como resultado base a superar.

Todas las pruebas realizadas sobre los dos reconocedores se explican con más profundidad en el Capítulo 4. El tiempo empleado en esta fase fue de 3 meses aproximadamente.

1.3.5 Diagrama de Gantt

En la Figura 1 se puede observar el diagrama de Gantt de este proyecto:

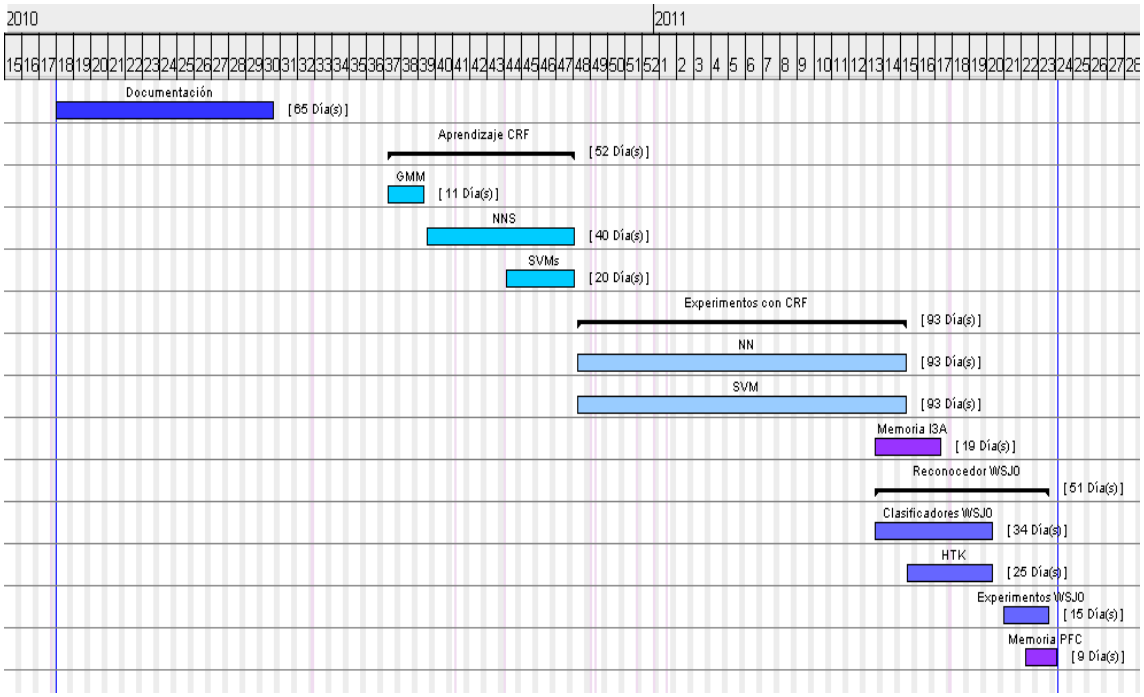

Figura 1. Diagrama de Gantt

2. Descripción

El reconocimiento automático del habla es básicamente un problema de clasificación estadística de secuencias de observaciones, donde queremos calcular la probabilidad a posteriori P(W|X). Donde W = $w_1w_2…w_m$ es la secuencia de palabras permitida por un diccionario y X = $x_1x_2…x_n$ es la representación paramétrica de la señal de voz. El problema de clasificación consiste en encontrar la secuencia de palabras *W* que maximice P(W|X). Para resolver este problema se utiliza el teorema de Bayes:

$$\widehat{W} = \arg\max_{w} P(W|X) = \arg\max_{w} \frac{P(X|W)P(W)}{P(X)} \qquad (2.1)$$

Dada una observación de la señal de voz X, en lugar de calcular P(W|X) podemos buscar aquella secuencia de palabras W que maximice:

$$\widehat{W} = \arg\max_{w} P(X|W)P(W) \qquad (2.2)$$

ya que la probabilidad de las observaciones P(X) no influye en la maximización. La probabilidad P(W) es conocida como el modelo del lenguaje (LM, language model) y la P(X|W) es conocida como el modelo acústico (AM, acoustic model). El objetivo es construir estos dos modelos de manera que reflejen de la manera más real posible el lenguaje hablado para poder reconocerlo. En sistemas de reconocimiento de gran vocabulario, hay una gran cantidad de palabras, de las cuales no podemos tener un modelo único, por lo que tenemos que dividirlas en secuencias de unidades más pequeñas como fonemas o trifonemas, por lo tanto P(X|W) está fuertemente relacionada con el modelo fonético. Además debe tener en cuenta la variabilidad de la voz, del género del hablante, de la pronunciación, del ambiente y también la variabilidad de la coarticulación en contexto dependiente de fonemas. El proceso de decodificación, donde se busca la secuencia de palabras W que mejor se adapte a la señal de voz X, en los sistemas de reconocimiento del habla es más que un simple problema de reconocimiento de patrones, puesto que en un sistema de reconocimiento de voz continua hay que buscar en una gran red de palabras que forman el modelo de lenguaje.

En este proyecto como ya se comentó en la introducción dejaremos por defecto el modelo de lenguaje y nos centraremos en mejorar el modelo acústico. A lo largo de este capítulo describiremos en el punto 2.1 un sistema de reconocimiento automático del habla estándar así como el estudiado en el presente proyecto, en los puntos 2.2 y 2.3 describiremos de forma más detallada las distintas fases del sistema.

2.1 Sistema de Reconocimiento Automático del Habla

La estructura estándar de un sistema de reconocimiento automático del habla (RAH) consta de tres etapas principales: En la primera, se extrae de la señal de voz sus características, obteniendo un vector de características compacto que intenta destacar de la señal sus rasgos más importantes. Existen varias técnicas para llevar a cabo la extracción de características que serán comentadas en el punto 2.2.1, que suelen consistir en un análisis localizado tiempo-frecuencia. En la segunda parte, estos vectores de características son las entradas al modelo acústico que es entrenado para asociar a cada vector una unidad con significado lingüístico. Esto se realiza entrenando el modelo acústico con los ejemplos de entrenamiento, que son los vectores de características, previamente etiquetados, obteniendo así un conjunto de modelos de mezclas de Gaussianas que modelan las distintas unidades lingüísticas. Y por último, las salidas del modelo acústico son probabilidades relativas que introducimos al decodificador para que obtenga la secuencia de palabras más probable. En la Figura 2 podemos observar la estructura de este sistema.

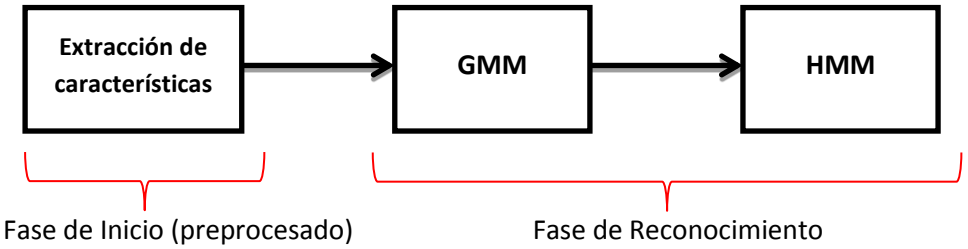

Figura 2. Sistema estándar de RAH

Como ya hemos comentado en el Capítulo 1 este sistema estándar llamado sistema GMM-HMM obtiene buenas prestaciones en numerosas tareas pero tiene algunas limitaciones que quieren ser eliminadas, y por ello se está utilizando desde hace unos años los sistemas Tandem. En el presente proyecto se ha estudiado un sistema Tandem como el que podemos ver en la Figura 3.

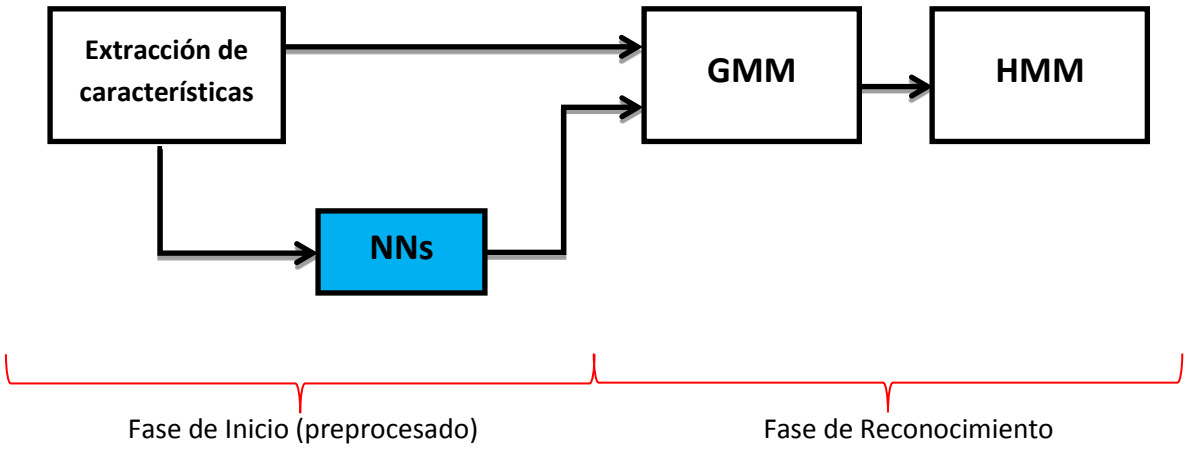

Figura 3. Estructura Sistema Tandem

Se puede observar en la Figura 3 que la diferencia que hay entre los dos sistemas es que en el sistema Tandem existe una etapa más. Esta nueva etapa consiste en un conjunto de clasificadores de rasgos fonéticos que en nuestro proyecto son redes neuronales, aunque pueden utilizarse otro tipo de clasificadores. Sobre estos clasificadores se hace una detallada descripción en el punto 2.2.2. La idea es obtener unos nuevos vectores de características que se utilicen como entradas al sistema GMM-HMM para mejorar el reconocimiento. Estarán formados por los vectores calculados en la etapa 1 y se les concatenarán las salidas obtenidas de los clasificadores de rasgos fonéticos.

Para describir con más detalle cada una de las etapas que forman el Sistema Tandem, hemos agrupado las etapas en dos fases: la fase inicial que la forman la etapa de extracción de características y los clasificadores de rasgos fonéticos, punto 2.2, y la fase de reconocimiento formada por las otras dos etapas restantes, punto 2.3.

2.2 Fase Inicial

La fase inicial de nuestro sistema Tandem consta de dos etapas: la extracción de características de la señal de voz y los clasificadores de rasgos fonéticos. A continuación describiremos cada una de estas etapas.

2.2.1 Extracción de características

Realizamos la extracción de características de la señal de voz para reducir su dimensionalidad y así poder eliminar información no relevante en el análisis fonético y realzar aquellos aspectos de la señal que contribuyen significativamente a su detección. Para llevar a cabo la extracción de características primero se realiza la adquisición de la señal de voz. En este proyecto, esta parte no tenemos que realizarla puesto que utilizamos las bases de datos TIMIT y WSJ0.

Existen un gran número de características, y todas ellas son características frecuenciales, como por el ejemplo, los coeficientes MFCC. Esto es así porque muchas de las características de la voz como son los formantes se caracterizan mejor en el dominio de la frecuencia con un vector de características de baja dimensión. Como hemos mencionado hay numerosos tipos de técnicas de parametrización de la señal de voz. En este proyecto haremos mención a las parametrizaciones utilizadas en nuestros experimentos, estas son: los coeficientes MFCC, lo más utilizados en RAH; los coeficientes PLP; y por último los coeficientes RASTA. Las tres técnicas utilizan el análisis localizado de la señal de voz, que consiste en enventanar la señal con una ventana normalmente de 25ms con un desplazamiento de 10ms, tratando estos fragmentos de la señal de manera independiente.

Para realizar la extracción de características en el presente proyecto se ha utilizado la herramienta OpenSmile, cuyo funcionamiento es explicado en el Anexo II.

Mel-Frecuency Cepstrum Coefficients (MFCC)

El cálculo de los coeficientes MFCC es explicado con detalle en el Anexo I. El esquema del proceso de obtención de los coeficientes MFCC se puede ver en la siguiente figura:

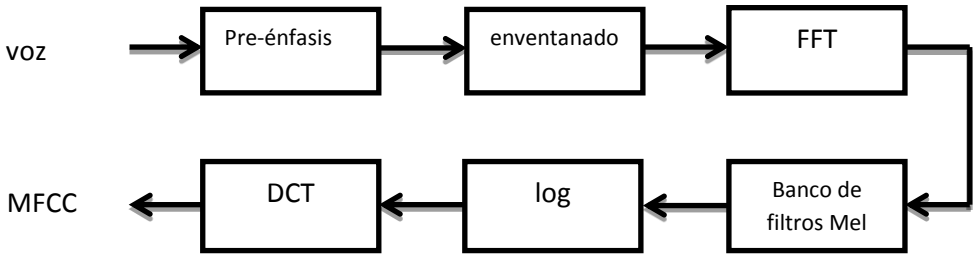

Figura 4. Esquema del proceso de extracción de características MFCC

De la Figura 4 podemos extraer que los coeficientes mel-cepstrum se obtienen de aplicar la DCT (Discrete Cosine Transform) a las salidas de los bancos de filtros Mel, realizando previamente un preprocesado de la señal de voz.

En el caso de reconocimiento de voz se suelen calcular los 13 primeros coeficientes MFCC. Nos interesa captar los cambios temporales que se dan en el espectro, ya que juegan un papel muy importante en la percepción humana y la coarticulación. Por ello se utilizan además los coeficientes delta ya que capturan esa información y los coeficientes de aceleración. Los coeficientes delta y los de aceleración no son más que la primera y segunda derivada de los coeficientes MFCC respectivamente. Por lo tanto se suele trabajar con un vector de características de 39 dimensiones.

Predicción Lineal Perceptual (PLP)

En la Figura 5 podemos ver el diagrama de bloques que se usa para realizar la extracción de los coeficientes PLP, la explicación detallada del proceso se realiza en el Anexo I.

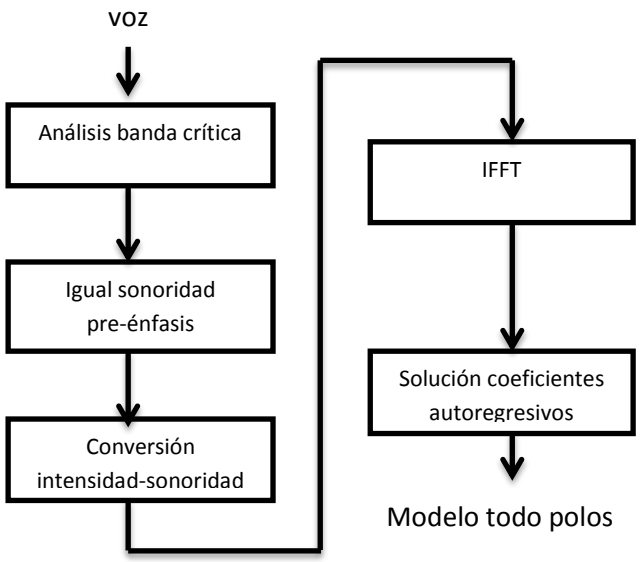

Figura 5. Diagrama de bloques de extracción de los coeficientes PLP.

El principio básico del análisis PLP es obtener mediante un modelo de todo polos una aproximación del espectro auditivo de la voz. El modelo de todo polos lo obtenemos mediante el método de autocorrelación y se utiliza la frecuencia Bark [5], a diferencia de los coeficientes MFCC que utilizan la frecuencia Mel [1].

El orden del modelo todo polos variará según el grado de detalle que queramos que tenga la aproximación del espectro auditivo de la voz.

Relative Spectral Transform (RASTA)

La técnica PLP es vulnerable cuando los valores del espectro localizado de la señal de voz se ven modificados por la respuesta en frecuencia del canal de comunicaciones, por ello aparece la técnica RASTA para hacer que el análisis PLP sea más robusto ante estas distorsiones espectrales [6].

El análisis RASTA consiste en reemplazar el espectro localizado por un espectro estimado en el que cada canal frecuencial es filtrado paso banda con un filtro que vale cero en la frecuencia cero, este cero sirve para eliminar las variaciones lentas o los valores constantes de cada canal frecuencial y también vale cero en frecuencias superiores a las utilizadas por la dinámica de la voz, normalmente decenas de Hz, para así eliminar variaciones demasiado rápidas. El nuevo espectro será menos sensible a variaciones lentas en el espectro localizado.

2.2.2 Clasificadores

En este punto hablaremos de los clasificadores de rasgos fonéticos cuyas salidas concatenaremos a los vectores de características comentados en el apartado anterior. El nombre de clasificadores de rasgos fonéticos se debe a que lo que vamos a clasificar es si las entradas tienen o no un determinado rasgo fonético, siendo las entradas los vectores de características que representan a la señal de voz. Vamos a realizar clasificadores binarios, es decir, vamos a tener un clasificador para cada rasgo fonético. Los rasgos fonéticos que queremos clasificar así como un resumen de lo que son, se encuentran explicados en detalle en el Capítulo 3.

En este proyecto se han investigado los siguientes clasificadores: modelos de mezclas de Gaussianas, redes neuronales artificiales y máquinas de soporte vectorial. A continuación realizaremos un breve resumen sobre cada uno de los clasificadores. En el Capítulo 3 describiremos como los hemos desarrollado y los experimentos realizados con ellos.

Modelos de Mezclas de Gaussianas (GMM)

Cuando utilizamos los GMM como clasificadores la idea básica consiste en modelar cada clase que queremos clasificar con una mezcla de Gaussianas. Para conseguirlo, primero entrenamos el modelo de mezclas con datos de entrenamiento, que son ejemplos de la clase que queremos clasificar, donde entrenar el modelo consiste en estimar sus parámetros.

Un vector aleatorio x que esta descrito por una mezcla de M Gaussianas tiene como función de densidad de probabilidad:

$$f(x) = \sum_{i=1}^{M} \lambda_i N(x|\mu_i, \sum_i) \qquad (2.3)$$

donde $N(x|\mu_i, \sum_i)$ es la función de distribución de una Gaussiana multidimensional con vector de medias μ_i y matriz de covarianzas $\sum_i$. La suma de todos los λ_i da uno y representa el peso que cada componente Gaussiana tiene en la mezcla.

Para calcular los parámetros $\lambda_i, \sum_i$ y μ_i utilizamos el criterio EM (Expectation Maximization) que es una solución iterativa que maximiza la verosimilitud utilizando una función auxiliar [1]. En general, la matriz de covarianzas de cada mezcla tiene N^2 elementos (donde N es la dimensión del vector de características) que tienen que ser estimados. Si las características del vector son independientes los elementos de fuera de la diagonal de $\sum_i$ son cero, por lo que solo se tienen que estimar N elementos, así que se buscará utilizar características que sean independientes. Una vez las mezclas hayan sido entrenadas podremos utilizarlas como clasificadores.

Redes Neuronales Artificiales (ANN)

Hay varios tipos de redes neuronales [9,10], pero en nuestro proyecto vamos a utilizar las redes neuronales MLP (Multilayer Perceptron) con alimentación hacia adelante, en la Figura 6 vemos un ejemplo de MLP. Estas redes usan para entrenarse el algoritmo back-propagation que es un algoritmo de gradiente descendente [1] que reduce el error cuadrático medio de la salida de una neurona. El entrenamiento de la red es un problema de optimización no lineal cuyo objetivo es encontrar un conjunto de pesos que minimicen la función de coste. La función de coste describe una superficie en el espacio de pesos, llamada superficie de error. Los algoritmos de entrenamiento pueden verse como métodos para encontrar el mínimo de la superficie. Normalmente una red consta de una capa de entrada, otra capa de salida y al menos una capa oculta. Se ha demostrado que con una sola capa oculta es suficiente para conseguir buenos resultados de clasificación, y en muchos problemas, se consiguen mejores convergencias [16]. Además este tipo de redes es capaz de aproximar cualquier función no lineal.

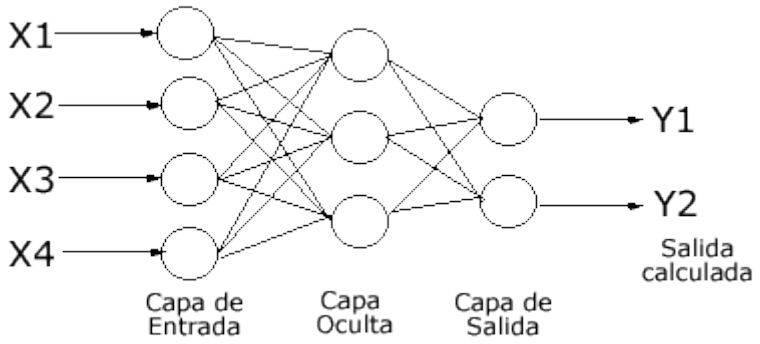

Figura 6. Red MLP, con una capa oculta.

Sea una red MLP con una capa oculta que tiene la siguiente estructura: la capa de entrada tiene p neuronas, la capa oculta tiene q neuronas y la capa de salida tiene r neuronas. El vector de salida Y está relacionado con el vector de entrada X de la siguiente manera:

$$Z_j = f_1\left(A_j + \sum_{i=1}^{p} W_{ij}X_i\right) \quad j = 1,2,\dots,q \qquad (2.4)$$

$$Y_k = f_2\left(B_k + \sum_{j=1}^{q} V_{jk}Z_j\right) \quad k = 1,2,\dots,r \qquad (2.5)$$

donde A y B son los vectores bias, W y V son los vectores de pesos y, f_1 y f_2 las funciones de activación de las neuronas de la capa oculta y de salida respectivamente. En este proyecto las funciones de activación que se han utilizado son: la función de Matlab tansig (hyperbolic tangent sigmoid transfer function) para la capa oculta y la función de Matlab logsig (logarithmic sigmoid transfer function) para la capa de salida.

Los pesos de la red neuronal tienen que ser inicializados. Este paso es uno de los más importantes para poder conseguir unos buenos resultados de clasificación.

Máquinas de Soporte Vectorial (SVM)

SVM se diseñó para ser un clasificador binario capaz de adivinar si un vector de entrada x pertenecía a la clase 1 (entonces la salida sería y=+1) o a la clase 2 (y=-1).

Dado un conjunto de datos separables, el objetivo del SVM es encontrar la función de decisión óptima, como se puede pensar hay un número infinito de funciones óptimas, en el sentido de obtener una solución sin ningún error. Lo que se busca es encontrar una función óptima que sea capaz de generalizar ejemplos no conocidos a priori. Podemos pensar en un algoritmo adicional para elegir entre aquellas funciones que nos den cero errores. Por ello la función que elegiremos será aquella que tenga el máximo margen, siendo este margen la distancia entre la muestra más cercana y el límite de decisión definido por la función. La elección de la función con máximo margen no implica obtener la solución con menos errores, pero en la mayoría de los casos esta elección es la mejor en muchos problemas [17].

En la Figura 7 se muestra un problema de clasificación para SVM:

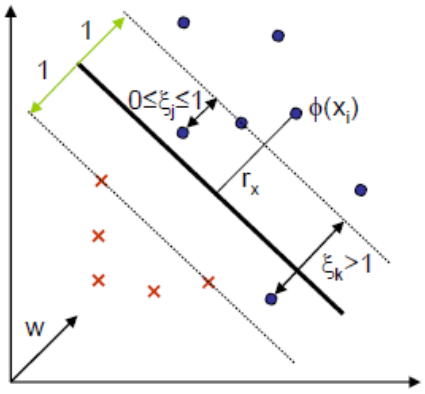

Figura 7. Margen de decisión.

El problema descrito en la Figura 7 se separaría con una función lineal. Muchos problemas de interés no se pueden separar tan fácilmente, por ello se define una función no lineal discriminante de la siguiente manera:

$$f(x_i) = w^T \phi(x_i) + b, \qquad (2.6)$$

donde $\phi(x_i): R^n \rightarrow R^{n'}, (n \ll n')$, es una función no lineal que mapea el vector x en un espacio, llamado espacio de características, de alta dimensionalidad incluso de hasta infinito. En este espacio de características se asume que las clases son linealmente separables. El vector w representa el hiperplano que divide este espacio. Debe tenerse en cuenta que el espacio de características que aquí mencionamos no es el mismo que el espacio de características de la señal de voz, sino que nos referimos al espacio donde se encuentran las entradas de una función Kernel, que será comentada más adelante.

En la Figura 7 vemos r_x que representa la distancia entre la muestra transformada $\phi(x_i)$ y el hiperplano de separación, y $\|w\|$ es la norma Euclidea de w. Los vectores más próximos al límite de decisión se llaman vectores soporte (support vectors) y son los que definen el margen, así como los únicos que utilizaremos para encontrar la solución. Por lo tanto para cada muestra x_i tenemos, $r_x = \frac{f(x_i)}{\|w\|}$. El objetivo es encontrar el clasificador que tenga la mínima $\|w\|$ cumpliendo la restricción de que todas las muestras estén bien clasificadas:

$$y_i = (w^T \phi(x_i) + b) \geq 1, \qquad (2.7)$$

podemos tratarlo como un problema de optimización cuadrática [3].

Para que el clasificador tenga una mejor capacidad de generalizar y de tratar con casos no separables, se le permite un número de ejemplos no clasificados, esto se consigue introduciendo una penalización.

Normalmente, la función $\phi(x_i)$ no es conocida pero esto no es un problema puesto que nosotros solo necesitamos evaluar el producto vectorial $\phi(x_i)^T \phi(x_j)$ el cual puede ser calculado utilizando la función Kernel $K(x_i, x_j)$.

Hay varios tipos de funciones Kernel, las más usadas son la lineal, la polimonial, RBF (función de base radial) y la sigmoide [24].

2.3 Fase de Reconocimiento

Como hemos dicho en secciones anteriores los sistemas de reconocimiento tienen dos fases: fase de inicio, donde realizamos la extracción de características y que ya ha sido explicada; y la fase de reconocimiento que va a ser explicada en esta sección y cuyas etapas corresponden con los bloques recuadrados en rojo de las Figuras 8, 9, 10 y 11.

En la Figura 8 podemos observar el sistema estándar utilizado en RAH:

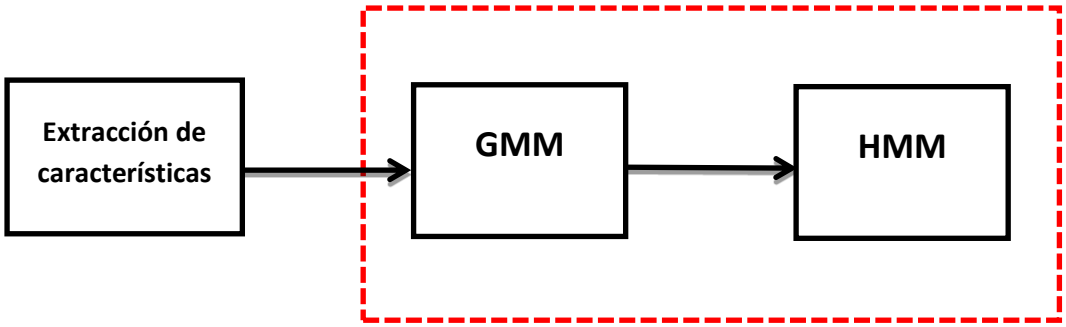

Figura 8. Sistema estándar en RAH

En este sistema estándar la fase de reconocimiento consta de dos etapas que constituyen el back-end: en la primera etapa (GMM) se realiza la estimación de los modelos. El número de modelos que tengamos dependerá de la exactitud que queramos que tenga nuestro reconocedor, podemos modelar palabras enteras, o como es habitual fonemas o trifonemas, estos últimos son los que mejor tasa de reconocimiento nos proporcionan. Cada modelo es un HMM (Modelo Oculto de Markov) y cada estado del HMM es modelado con una mezcla de Gaussianas de ahí el nombre del bloque. La segunda y última etapa (HMM) consiste en la decodificación de los HMM, es decir, en encontrar la secuencia de palabras más probable dada la entrada que tenemos en el sistema.

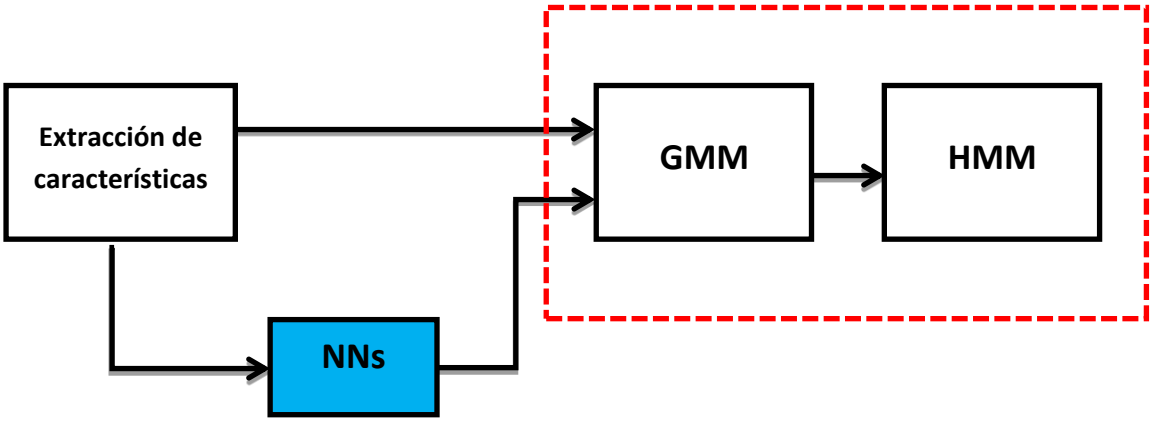

Figura 9. Introducción de redes neuronales en el sistema estándar RAH.

La primera modificación que se introduce en el sistema de la Figura 8 se puede ver en el sistema RAH de la Figura 9. Realmente esta modificación afecta a la fase de inicio, donde la fase de reconocimiento, que es la misma que en el sistema estándar, sus entradas no sólo son las características comunes sino también las salidas de los clasificadores de rasgos fonéticos que son implementados con redes neuronales. Lo que se espera realizando esta modificación es mejorar los resultados del reconocedor ya que se está introduciendo información adicional que puede ser de ayuda a la hora de reconocer. Realmente esto no ocurre debido a que las

salidas de los clasificadores de rasgos fonéticos no son Gaussianas, como en el caso de las salidas del bloque de extracción de características, y por lo tanto se necesita adaptar las salidas de los clasificadores a los modelos para que así se pueda utilizar su información de forma eficaz.

Para adaptar las salidas de las redes neuronales a los modelos, vamos a utilizar la de-correlación PCA [27] que es el nuevo bloque que se añade al sistema RAH, Figura 10:

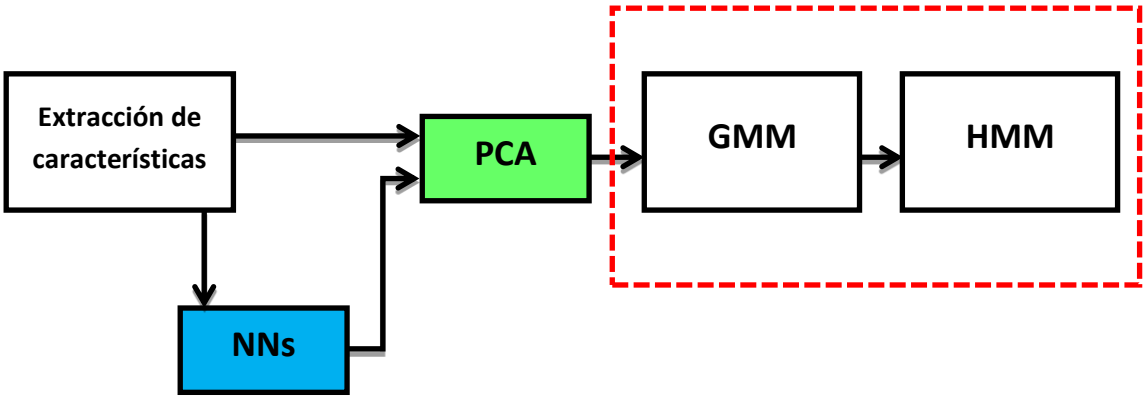

Figura 10. Sistema RAH con de-correlación PCA.

Al introducir este nuevo bloque conseguimos mejorar las prestaciones del reconocedor que era el objetivo de añadir los clasificadores de rasgos fonéticos a la fase de inicio.

Por último, se plantea el sistema RAH de la Figura 11:

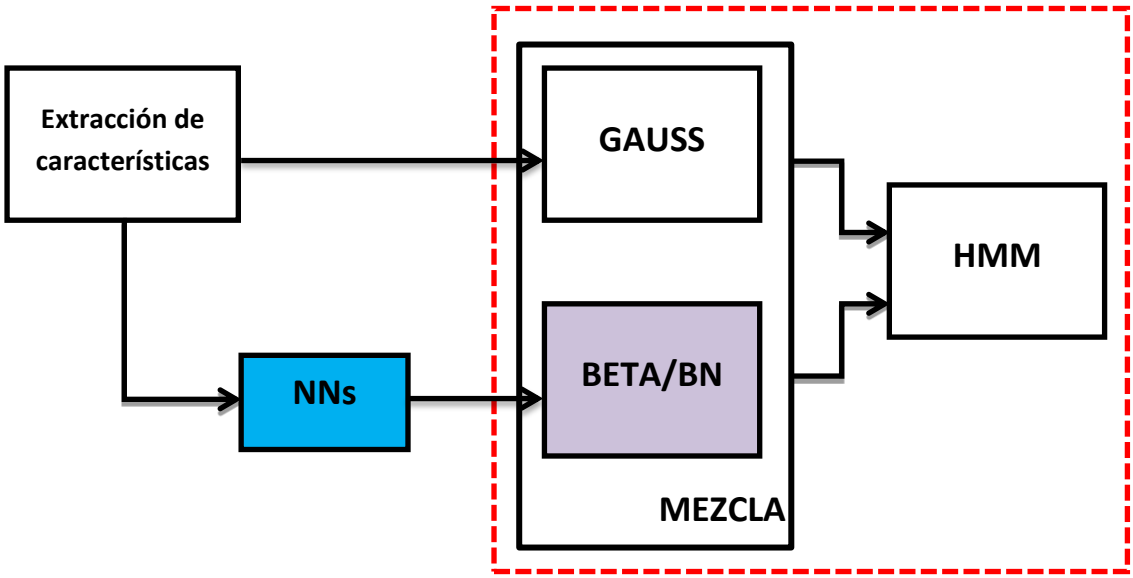

Figura 11. Sistema RAH que añade a su fase de reconocimiento distribuciones Beta o redes Bayesianas para crear una parte de los modelos.

Este nuevo sistema se presenta como una novedad en el estado del arte de los sistemas de reconocimiento automático del habla de gran vocabulario.

La novedad que se plantea es, que en lugar que las salidas de las redes neuronales se adapten a los modelos de mezclas de Gaussianas, estas se utilicen como observaciones de otro tipo de modelos a los que si se adaptan sin tener que hacer ningún paso previo como el de PCA. Se plantean tres posibles soluciones: redes Bayesianas, distribución Beta y distribución de Bernoulli. Las salidas de las redes si están adaptadas a la distribución Beta, puesto que son valores continuos que están entre 0 y 1. Cuando se utilicen las redes Bayesianas y la distribución de Bernoulli las salidas se cuantificarán y se convertirán en valores discretos de valor 0 o 1.

3. Clasificadores de rasgos fonéticos

3.1 Rasgos fonéticos

La Fonética es la disciplina que estudia los sonidos desde el punto de vista de su producción, transmisión y percepción, sin preocuparse del significado de los mismos. Dentro de la fonética se puede distinguir la articulatoria y la acústica. En la articulatoria se estudia el movimiento de los órganos fonadores para la formación y emisión del sonido; y la acústica se preocupa de las características de la onda sonora y su percepción. Nosotros nos centraremos en la fonética acústica.

Cuando se habla de los rasgos fonéticos de un sonido, lo que se quiere decir es cuales son las características más relevantes de ese sonido. Una manera de obtener esas características es estudiar los sonidos desde un punto de vista espectral. Una vez se sepa cuáles son sus características se podrían utilizar para clasificar los distintos sonidos.

Antes de seguir avanzando debemos aclarar que los fonemas son la unidad fonológica más pequeña, no tienen significado por sí mismos, pero si cambian el significado de una palabra si intercambiamos dos fonemas. Cada lengua o idioma posee un número distinto de fonemas así como algunos fonemas propios de su lengua.

Los sonidos se pueden dividir fundamentalmente en dos tipos: sonoros y no sonoros. Las vocales, por ejemplo, son sonidos sonoros y las consonantes hay tanto sonoras como no sonoras. La diferencia fundamental entre sonidos sonoros y sordos (no sonoros) es que los sonoros siguen un patrón regular en su estructura frecuencial y típicamente tienen más energía que los sordos, debido a la excitación de las cuerdas vocales. Para mostrar esta diferencia, se va a poner como ejemplo la palabra ‘she’ (ella en inglés) que está formada por el fonema /sh/ que es fricativo y sordo, y el fonema /ix/ que es vocálico. Este ejemplo se puede ver en la Figura 12:

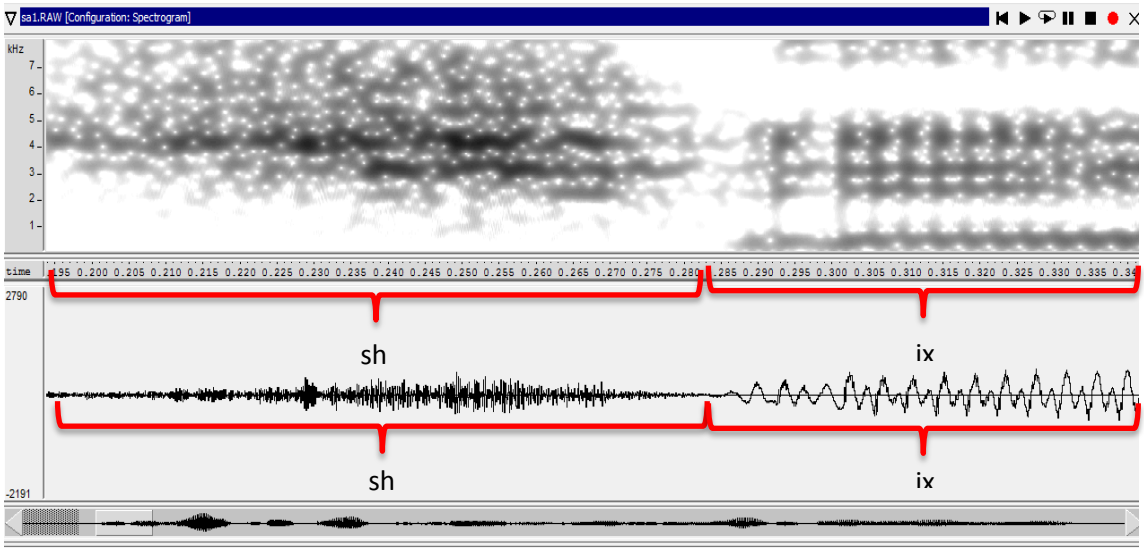

Figura 12. La imagen de arriba corresponde con el espectrograma de la palabra 'she' compuesta por los fonemas /sh//ix/, y la imagen inferior corresponde con su representación temporal. Para obtener estas gráficas se ha utilizado el programa WaveSurfer.

Los sonidos se dicen que son sonoros cuando se produce una vibración de las cuerdas vocales que en los sonidos sordos no se produce. Por eso cuando se observa una señal en tiempo de un sonido sonoro se puede ver cierta periodicidad que es debida a esa vibración y los sonidos sordos no tienen ninguna periodicidad, son más parecidos a una señal de ruido, Figura 12.

A parte de poder dividir los sonidos entre sonoros y sordos, se pueden realizar otras clasificaciones, como por ejemplo, sonidos vocálicos o fricativos que han sido nombrados antes. A continuación se van a estudiar diferentes tipos de sonidos: vocálicos, nasálicos, fricativos, africados, oclusivos o explosivos y líquidos.

Vocales

Se comenzará analizando los sonidos vocálicos, la característica principal que tienen se basa en unas estructuras de formantes claras, debido a la emisión de flujo de aire por el conducto bucal sin apenas resistencia, y con las cavidades resonadoras potenciando los armónicos distintivos de cada vocal. En la Figura 13 en la parte seleccionada con las dos líneas rojas se puede ver claramente las concentraciones de energía estables en el tiempo (líneas horizontales) que corresponden con los cuatro formantes del fonema vocálico /ah/.

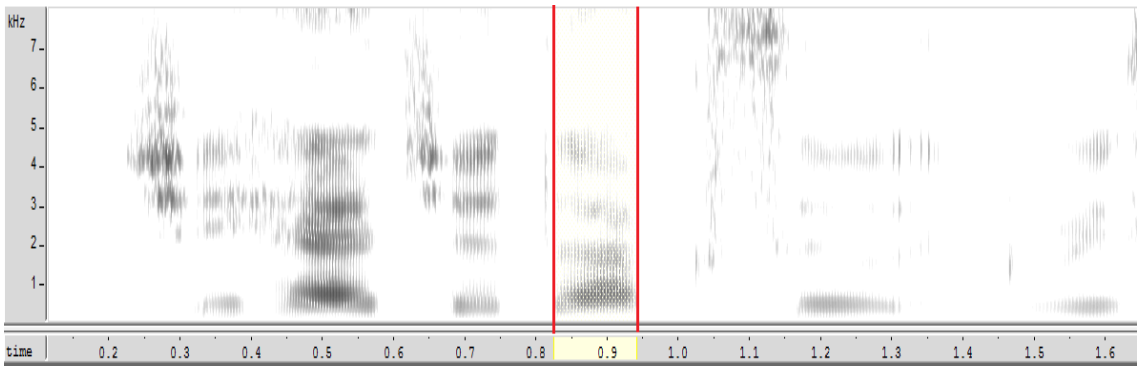

Figura 13. Espectrograma de la frase “She had your dark suit in greasy wash water all year”

Oclusivas

Los sonidos explosivos u oclusivos se producen por el cierre u oclusión de los órganos fonadores durante un intervalo de tiempo, seguido de su apertura con la correspondiente salida brusca de aire (explosión). Existen dos tipos de sonidos explosivos: los sordos y los sonoros. Suelen ser fáciles de observar en la señal temporal porque normalmente corresponden a una primera parte de silencio y de repente señal (como una explosión), Figura 14.

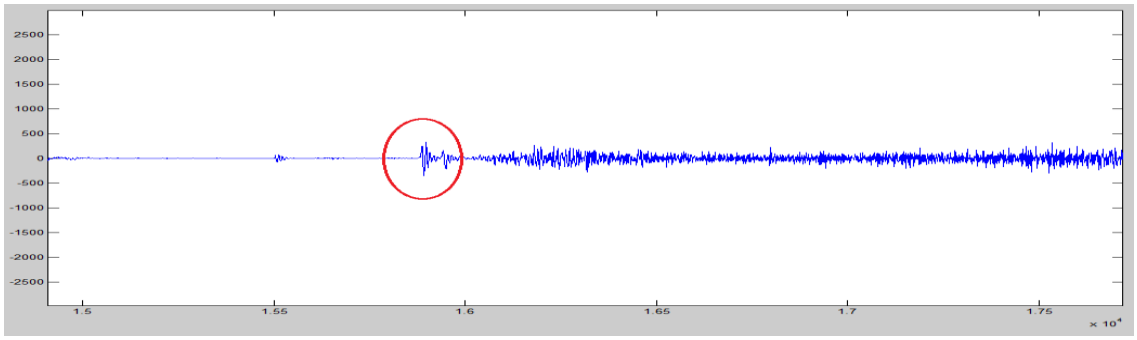

Figura 14. Representación temporal de una señal de voz. La explosión que se produce al abrirse los órganos fonadores se puede observar en la parte rodeada en rojo.

También son fácilmente detectables en un espectrograma, puesto que en la parte del silencio hay poca energía en todas las bandas durante su duración, como se puede observar en la Figura 15:

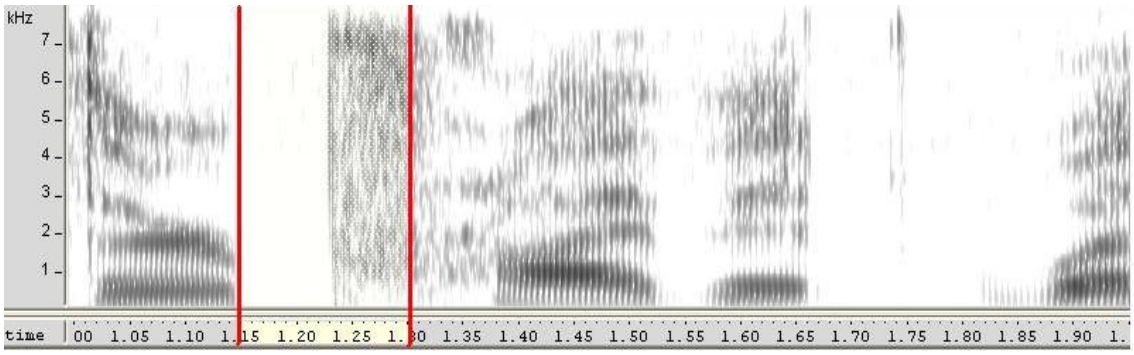

Figura 15. Espectrograma del fonema oclusivo /p/ (parte seleccionada en rojo) en la palabra "private".

Nasales

Para tener un sonido nasal se tiene que producir el cierre de los órganos articulatorios bucales, con la consiguiente expulsión del aire a través de los conductos nasales. La cavidad nasal se comporta como un resonador en paralelo con el tracto vocal, por lo que absorbe ciertas frecuencias, que como podemos ver en la Figura 16, absorbe las frecuencias más altas.

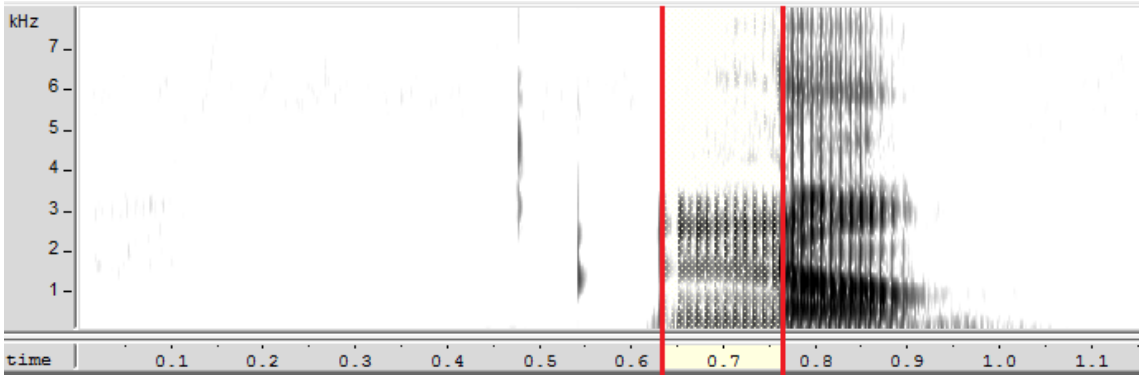

Figura 16. Espectrograma de la palabra "no". La parte recuadrada de rojo corresponde al fonema /n/ y la otra parte al fonema /o/.

Fricativas

Los sonidos fricativos se producen cuando se realiza un estrechamiento entre dos órganos articulatorios produciéndose la fricación. Las fricativas se suelen diferenciar del resto porque poseen muy altas frecuencias, no tienen una estructura periódica y presentan unas turbulencias que se comportan de forma similar a una señal de ruido, como se puede ver en la Figura 17.

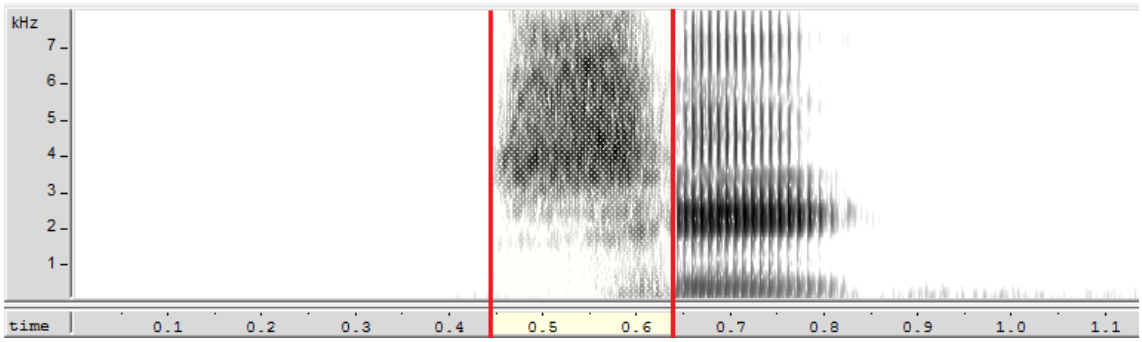

Figura 17. Espectrograma de la palabra “si”. La primera parte que esta recuadrada en rojo corresponde con el fonema /s/ y la segunda parte con el fonema /i/.

Africadas

Un sonido africado es una consonante que empieza como una oclusiva, pero que al soltar el aire se convierte en una fricativa. Las africadas se comportan como un estado intermedio entre oclusivas y fricativas, pero fonéticamente son secuencias de oclusivas más fricativas. En la Figura 18, en la parte seleccionada en rojo, podemos ver que la primera parte corresponde con el silencio que tienen todas las oclusivas al principio y que la segunda parte del espectro es como el de una fricativa, toda la energía está contenida en las altas frecuencias.

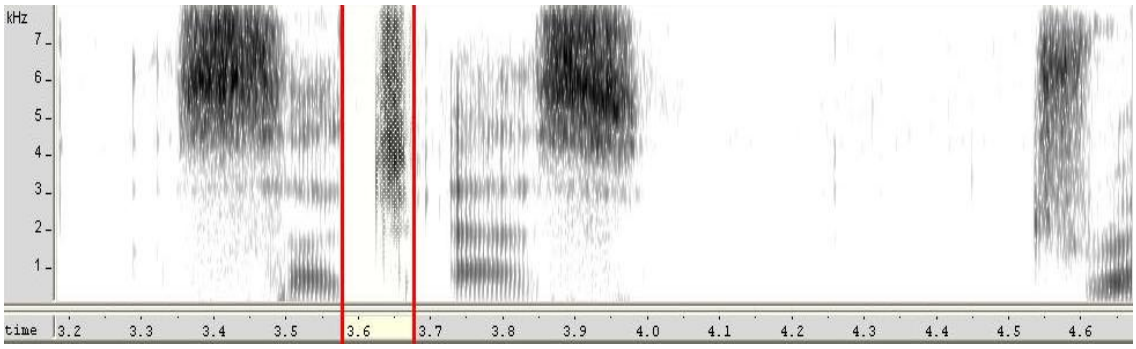

Figura 18. Espectrograma del fonema africado /ch/ (parte seleccionada en roja) en la palabra “such”.

Líquidas

Los sonidos líquidos son los más parecidos a las vocales, se articulan con el tracto abierto, como las vocales, y aunque existe algún obstáculo, dicho obstáculo no impide la salida de aire por los espacios que deja libre. En las Figuras 19 y 20 se puede ver el parecido que hay entre un sonido líquido y otro vocálico.

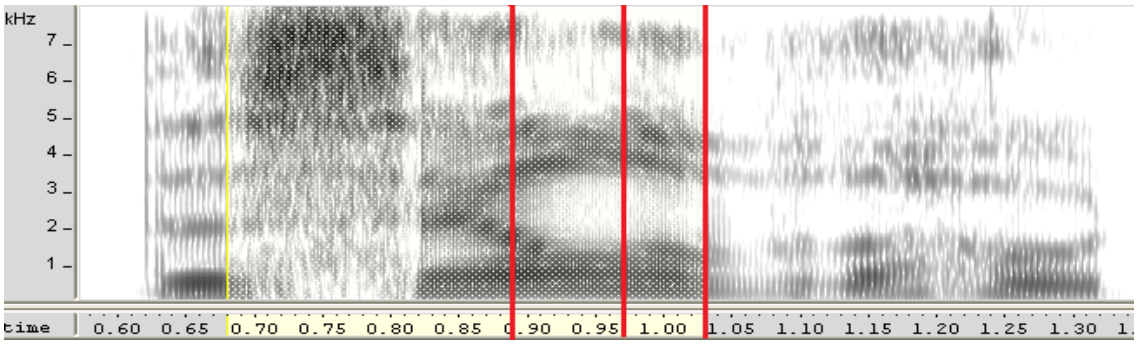

Figura 19. Espectrograma de los fonemas: /l/ y /ah/ respectivamente (partes seleccionadas en rojo) en la palabra “sale”.

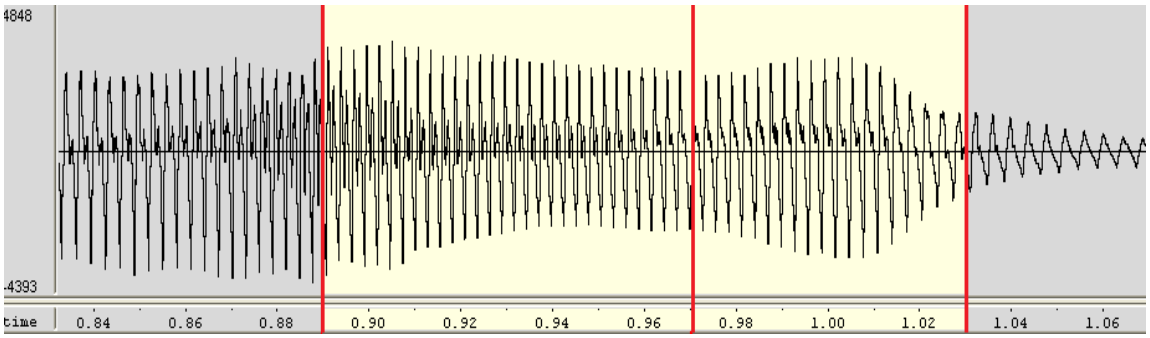

Figura 20. Representación temporal de los fonemas: /l/ y /ah/ respectivamente (partes seleccionadas en rojo) en la palabra “sale”.

3.2 Desarrollo de los clasificadores

Una vez que se han comentado las características de algunos rasgos fonéticos, el siguiente paso es realizar los clasificadores de estos rasgos. Pero para ello primero se tiene que aprender a utilizar los distintos tipos de clasificadores, que son: la red neuronal y la máquina de soporte vectorial (SVM). Después, se llevará a cabo una serie de experimentos en el punto 3.3 para poder elegir qué clasificador se utilizará.

En esta sección, por lo tanto, se hablará primero de las redes neuronales en el punto 3.2.1, y después de los SVM en el punto 3.2.2, en las dos ocasiones será desde un punto de vista práctico, puesto que la teoría sobre la que se basan los dos clasificadores ya ha sido introducida en el Capítulo 2.

3.2.1 Redes Neuronales

Para poder comprender más rápidamente el funcionamiento de las redes neuronales se comenzó realizando un experimento sencillo, que consiste en implementar la XOR.

La red neuronal se va a implementar utilizando Matlab, este posee una toolbox para redes neuronales que utilizaremos. En el Anexo III se encuentra un ejemplo de cómo se puede utilizar esta toolbox.

Ejemplo sencillo XOR

En la Tabla 1 podemos ver la tabla de verdad de la XOR:

Entrada A	Entrada B	Salida
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 1. Tabla de verdad puerta XOR

De la Tabla 1 podemos deducir que se van a clasificar dos clases: clase 0 y clase 1, por lo que vamos a tener un clasificador binario, que es el tipo de clasificadores que queremos implementar para cada rasgo fonético, como se verá en el punto 3.3.

Generamos un corpus artificial de 2000 ejemplos, de forma que podamos estudiar su clasificación en estas dos clases. Para añadir incertidumbre al problema de clasificación añadimos ruido aleatorio a las observaciones de la Tabla 1, Figura 21:

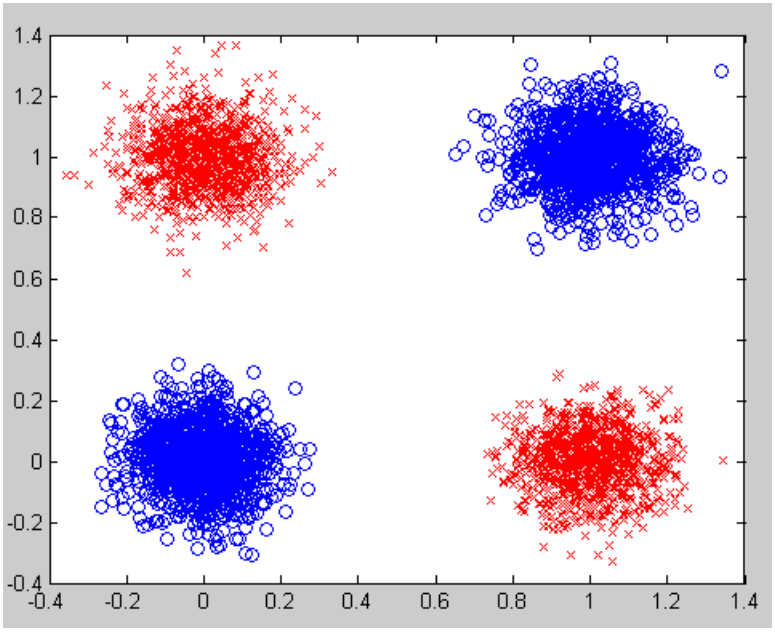

Figura 21. Datos generados aleatoriamente para la implementación de la XOR.

Los puntos de color rojo deberían ser clasificados dentro de la clase 1 y los de color azul dentro de la clase 0.

La idea es que hay que entrenar la red neuronal con unos datos de entrenamiento que tienen que estar etiquetados, es decir, la red neuronal tienen que saber a qué clase pertenece cada dato de entrenamiento. Este etiquetado se realiza manualmente. La manera de introducir los datos a la red neuronal puede hacerse de dos maneras en Matlab: con la función train(), se cargan todos los datos en memoria y se entrena la red; y con la función adapt(), se van introduciendo los datos poco a poco, en bloques. Los datos que se introducen en la red con la función adapt() tienen que estar balanceados, es decir, intercalar datos de una clase con los de la otra. Se van a utilizar estas dos funciones de la siguiente manera: se crea la red neuronal y se entrena con una pequeña parte de los datos con la función train(), es una manera de inicializar la red, y por último con adapt() se va entrenando la red con bloques de datos hasta que se hayan introducido todos nuestros datos de entrenamiento, en cada iteración adapt() nos devuelve la red actualizada, las salidas de la red y el error cometido. La razón de realizar así el entrenamiento de la red neuronal es porque para los experimentos de nuestro proyecto la base de datos de voz que se usa es enorme, por lo que se tendrá un problema de memoria si se hace solo con la función train().

A la hora de realizar el test, se tienen dos posibilidades: realizar validación cruzada, es decir, se puede dividir el conjunto de datos de entrenamiento en dos o más grupos, si lo dividimos en dos, primero entrenaremos la red con un grupo y la testearemos con el otro pero también lo haremos al revés, es decir, los datos que hemos utilizado en el entrenamiento antes serán los de test y los de test de antes serán ahora los de entrenamiento. Los dos resultados que se obtengan se promedian y así obtenemos la tasa de acierto del clasificador. La segunda opción es utilizar un conjunto de datos de test, diferente al de entrenamiento. En este ejemplo vamos a generarnos un conjunto de test de la misma manera que se ha hecho para los datos de entrenamiento, Figura 22:

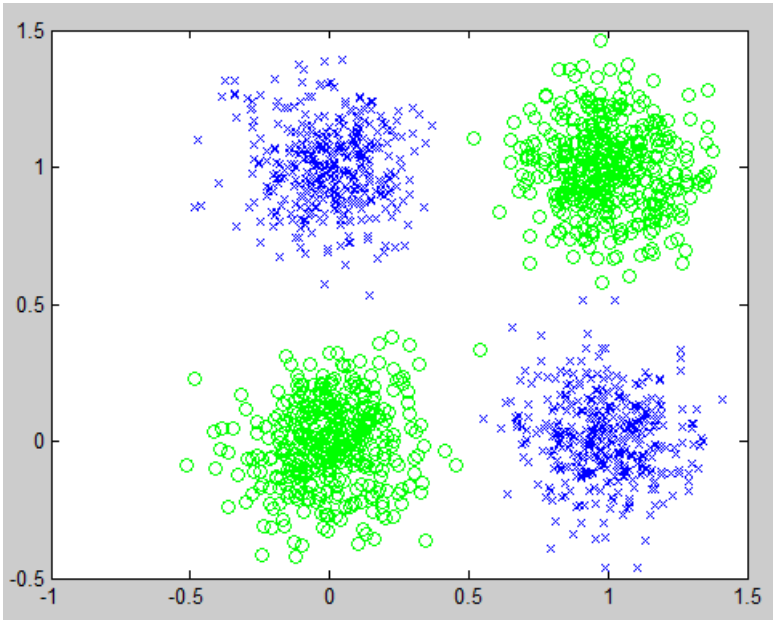

Figura 22. Conjunto de datos de test (800 ejemplos)

En el Capítulo 2 ya se ha dicho que con una sola capa oculta la red neuronal consigue muy buenos resultados, por lo que se van a realizar todos los experimentos con 3 capas: capa de

entrada, capa oculta y capa de salida. La capa de salida siempre va a tener dos neuronas ya que se quieren clasificar dos clases, el número de neuronas de la capa de entrada y oculta será el mismo, y este se irá modificando para ver cómo afecta a la clasificación.

En la Tabla 2 podemos ver la tasa de acierto conseguida con distintos valores para el número de neuronas de la capa de entrada y oculta:

Neuronas Capa de Entrada	Neuronas Capa Oculta	Tasa de Acierto en el entrenamiento (%)	Tasa de Acierto (%)
2	2	99.97	73.50
3	3	70.42	74.38
4	4	74.75	72.63
5	5	100.00	99.32
6	6	100.00	98.69
7	7	75.00	75.00
8	8	**100.00**	**99.50**
9	9	68.05	67.50
10	10	68.62	66.43

Tabla 2. Resultados de clasificación para XOR.

La mejor tasa la obtenemos con 8 neuronas. Se podía pensar que la red neuronal obtiene mejor resultado a mayor número de neuronas en las capas de entrada y oculta, pero como se puede observar en la Tabla 2, con 8 conseguimos la mejor tasa pero al poner una neurona más la tasa decae bastante. También debemos fijarnos en la tercera columna, esta tasa la obtenemos de las salidas que nos devuelve adapt(), en la mayoría de los casos es mayor que la tasa de acierto que obtenemos con test que es la que realmente nos interesa y además si nos fijamos en la primera fila, donde el número de neuronas es 2 la tasa de entrenamiento nos da un 100% de acierto y la de test un 73.5%. Por lo tanto, como conclusión se puede decir que no nos podemos fiar al 100% de la tasa que obtenemos solo con los datos de entrenamiento (3 columna de la Tabla 2), y que poner un número elevado de neuronas en la capa de entrada y oculta no nos asegura mejorar la clasificación. Este problema es muy conocido en el ámbito de reconocimiento de patrones, se conoce como sobreajuste u overfitting, consiste en que si se utiliza un número pequeño de datos la red se acaba aprendiendo esos datos no una generalización de estos. Debemos añadir también que a mayor número de neuronas mayor es el coste computacional.

3.2.2 Máquinas de soporte vectorial (SVM)

En el caso de los SVM la herramienta que se ha utilizado para trabajar con ellos ha sido el programa WEKA versión 3-6-3 de la universidad de WAIKATO, Halminton, Nueva Zelanda; con la librería LibSVM para máquinas de soporte vectorial. Con este programa se puede trabajar tanto por línea de comandos como de una forma gráfica. Se utilizó en primer lugar su forma gráfica para realizar pruebas con ejemplos sencillos, ya que es más intuitiva, pero para los experimentos de nuestro proyecto se utilizó la línea de comandos, por ser más rápida.

Los datos que utiliza WEKA están en formato Arff, por lo que no se pueden introducir los datos que se utilizan en la red neuronal tal y como están, tienen que ser convertidos al formato .arff. En nuestro caso tendremos que escribir las características de la voz en este formato, para ello se utilizó el programa OpenSmile que calcula una gran variedad de parametrizaciones y las escribe en varios formatos, uno de ellos es Arff, esto se realiza en los experimentos del punto 3.3.

Como se explicó en el Capítulo 2 el clasificador SVM puede utilizar varios tipos de funciones Kernel, para estos experimentos sencillos y para el resto de experimentos desarrollados en este proyecto se va a utilizar siempre la función de base radial (RBF) [18]. Por lo tanto los parámetros que van a influir en la clasificación y que se deberán modificar son: C parámetro de penalización y Υ parámetro de la función Kernel RBF.

Ejemplo Sencillo

Para realizar este ejemplo sencillo que nos sirve como entrenamiento para experimentos grandes se utilizarán los datos Iris.arff, datos que son proporcionados por el propio programa WEKA, y son muy utilizados [25].

Se quieren clasificar los datos en 4 clases: sepallength, sepalwidth, petallength y petalwidth. Los datos de entrenamiento se pueden ver en la Figura 23. Para realizar el test se pueden elegir dos opciones, igual que en el caso de la red neuronal: validación cruzada o utilizar un conjunto de datos de test. Para este ejemplo se utilizará la opción de validación cruzada.

Una vez que se tienen los datos, el paso siguiente es introducir el valor de los parámetros en el clasificador SVM; y así ya se puede realizar la clasificación.

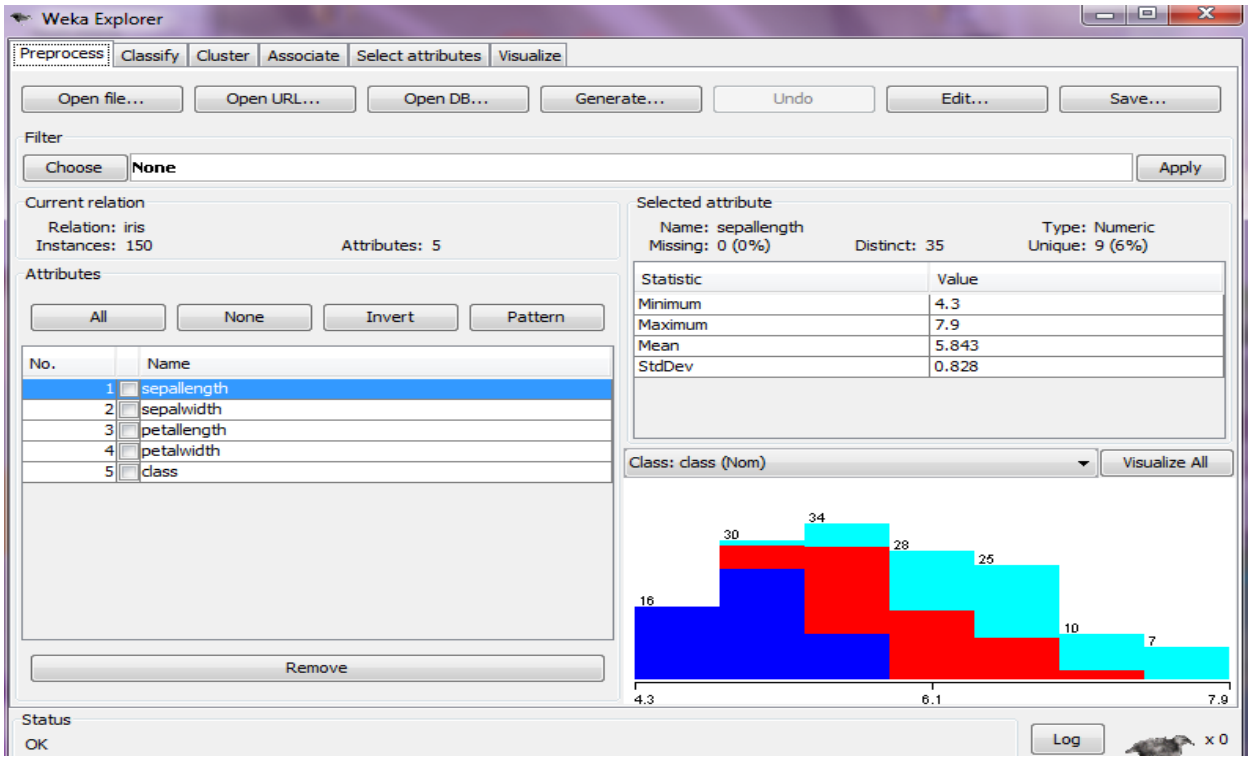

Figura 23. Programa WEKA. Distribución de los datos Iris.arff.

En la Figura 24 se puede observar la información que nos proporciona el clasificador SVM en WEKA, de la cual el de mayor importancia para nosotros es el porcentaje de datos bien clasificados.

```
=== Summary ===

Correctly Classified Instances         145               96.6667 %
Incorrectly Classified Instances         5                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0222
Root mean squared error                  0.1491
Relative absolute error                  5      %
Root relative squared error             31.6228 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        Iris-setosa
                 0.94      0.02       0.959     0.94      0.949      0.96     Iris-versicolor
                 0.96      0.03       0.941     0.96      0.95       0.965    Iris-virginica
Weighted Avg.    0.967     0.017      0.967     0.967     0.967      0.975

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Figura 24. Información que nos devuelve el clasificador SVM.

Una vez que ya se sabe cómo funcionan los clasificadores, se pasará a explicar los experimentos que se han realizado para el presente proyecto en el punto 3.3, que consisten en realizar una serie de clasificadores de rasgos fonéticos, que nos servirán como ejemplos para desarrollar los clasificadores para la base de datos WSJ0 que es la que vamos a introducir en el reconocedor.

3.3 Experimentos

Para llevar a cabo los experimentos se va a utilizar la base de datos TIMIT. Este punto lo dividiremos de la siguiente forma: primero una breve descripción sobre TIMIT y por último, descripción y evaluación de los diferentes experimentos realizados.

3.3.1 TIMIT

La base de datos TIMIT [21] ha sido diseñada para utilizar sus datos en la adquisición de conocimiento fonético y acústico de la voz y para desarrollar y evaluar sistemas de reconocimiento automático del habla.

Se crea bajo el esfuerzo conjunto de varios grupos, respaldados por el *Defense Advanced Research Projects Agency - Information Science and Technology Office* (DARPA-ISTO). Los datos de test fueron diseñados entre el Instituto Tecnológico de Massachusetts (MIT), el Instituto de Investigación de Stanford (SRI), y Texas Instruments (TI).

TIMIT contiene un total de 6300 frases, pronunciadas por 630 hablantes, donde cada uno de ellos pronuncia 10 frases. De los 630 hablantes: 438 son hombres y 192 son mujeres, todos ellos pertenecen a una de las 8 principales regiones con dialecto que hay en Estados Unidos, por ellos los ejemplos tanto de entrenamiento como de test se encuentran divididos en 8 carpetas, una por cada uno de los dialectos: dr1,dr2,..dr8.

3.3.2 Experimentos realizados

El objetivo de los experimentos es encontrar cuál de los clasificadores descritos en el punto 2.2.2 es el que se utilizará en nuestros experimentos finales con el reconocedor. Vamos a probar dos tipos de clasificadores: la red neuronal y la máquina de soporte vectorial (SVM), y compararemos los resultados obtenidos. Una vez hayamos comparado los resultados, elegiremos aquel clasificador que mejor cumpla los requisitos requeridos. Los requisitos que tiene que cumplir el clasificador son dos: máxima tasa de acierto y coste computacional bajo.

Vamos a clasificar dos rasgos fonéticos: sonido nasal y oclusivo. Para cada rasgo implementaremos un clasificador binario. Las entradas a los dos clasificadores son los vectores de características. Vamos a dividir el conjunto de entrada en dos, es decir, si queremos clasificar que el vector de características sea un sonido oclusivo, tendremos el conjunto de ejemplos que son oclusivos, etiquetados manualmente, y el conjunto de ejemplos que no son oclusivos, también etiquetados manualmente. Esta división del conjunto de entrada es una

fase de preparación de los datos para el clasificador. Una vez realizada esta separación de los datos de entrada lo que tenemos son vectores de características con una dimensión de 65, este número proviene de los 13 coeficientes estáticos de los coeficientes MFCC o PLP, ya que dejaremos los coeficientes RASTA para futuros experimentos, y los 52 restantes son los 13 coeficientes estáticos de las dos tramas anteriores y las dos posteriores, que concatenamos para capturar el contexto y la coarticulación, en sustitución de las derivadas. Hay varios tipos de coeficientes MFCC y PLP, para nuestros experimentos vamos a utilizar los siguientes:

- MFCC_0,incluimos el coeficiente cero dentro de los 13 coeficientes estáticos
- PLP_0, incluimos el coeficiente cero dentro de los 13 coeficientes estáticos.

A continuación se pasará a explicar los experimentos realizados con cada uno de los clasificadores.

Red Neuronal

Para realizar los clasificadores vamos a utilizar un tipo de red neuronal concreto, la MLP (Multilayer perceptron). La estructura de la red neuronal consta de: una capa de entrada, una capa de salida y una capa oculta. La capa de salida está formada por dos neuronas, ya que es un clasificador binario, y el número de neuronas de la capa de entrada y salida va a ser un parámetro que vamos a modificar para poder conseguir mejores resultados en la clasificación, como se comentó en el punto anterior.

Por simplicidad vamos a imponer que el número de neuronas de la capa de entrada y de la capa oculta sea el mismo, el valor de este parámetro influye mucho en el coste computacional, por lo que aumentar el número de neuronas de la capa implica aumentar el coste computacional. Por ello el rango elegido de valores con el que vamos a experimentar es de 10 neuronas hasta 20 para la capa de entrada y la capa oculta.

En las Tablas 3 y 4 recogemos los mejores resultados obtenido de los 11 posibles experimentos, para el clasificador de nasales y de oclusivas respectivamente:

Número de neuronas Capa de entrada	**Número de neuronas Capa oculta**	**Características**	**Tasa de acierto (%)**
18	18	MFCC_0	73.48
18	18	PLP_0	**76.20**
16	16	PLP_0	74.53
12	12	PLP_0	71.79

Tabla 3. Resultados de la clasificación de nasales.

Número de neuronas Capa de entrada	Número de neuronas Capa oculta	Características	Tasa de acierto (%)
12	12	PLP_0	66.95
14	14	PLP_0	**77.70**
18	18	PLP_0	68.73
20	20	PLP_0	68.38

Tabla 4. Resultados de la clasificación de oclusivas.

La máxima tasa de acierto para ambos clasificadores la conseguimos utilizando los coeficientes PLP_0, para nasales la tasa máxima es de 76.20 % y para el clasificador de oclusivas es de 77.70 %. Podemos destacar en la clasificación de oclusivas que el valor del número de neuronas influye considerablemente en la tasa de clasificación, también en la clasificación de nasales pero no tan fuertemente.

Máquina de soporte vectorial (SVM)

El otro tipo de clasificador que vamos a utilizar es el SVM. Como se explicó en el punto 2.2.2 para realizar la clasificación con los SVM necesitamos una función de Kernel, la función de Kernel que hemos elegido para trabajar es la RBF (Función de base radial) [18]:

$$K(x, y) = e^{-\gamma\|x-y\|^2} \qquad (3.1)$$

La pareja (C, γ) son los parámetros a cambiar, donde γ es el parámetro de la función RBF y C es el parámetro de penalización. Los valores que vamos a dar a cada uno de ellos son [18]:

- Para γ = 2^{-15}, 2^{-13},…, 2^{3}
- Para C =2^{-5}, 2^{-3},…, 2^{15}

Vamos a entrenar el clasificador SVM con cada una de las posibles combinaciones (C, γ), que suman un total de 110 posibles combinaciones. Para el entrenamiento no vamos a utilizar todo el conjunto de entrenamiento de TIMIT, como ya se comentó los clasificadores SVM no pueden manejar gran cantidad de datos, por ello vamos a utilizar solo el 5% del conjunto de entrenamiento, elegido aleatoriamente de la siguiente forma:

DR1	DR2	DR3	DR4	DR5	DR6	DR7	DR8
8 %	16 %	16 %	16 %	16 %	7 %	16 %	5 %

Tabla 5. Porción de datos elegida de cada una de las carpetas de la base de datos TIMIT para el conjunto de entrenamiento y test.

Haremos lo mismo para el conjunto de test que se utilizará para seleccionar la pareja (C, γ) que mejor tasa de acierto nos proporciona. Una vez seleccionada esa pareja (C, γ) pasaremos a utilizar todo el conjunto de test con el clasificador. De esta forma nos aseguramos de que la elección de los parámetros no se sobre ajusta a los datos de test.

En la Tabla 6 y 7 podemos ver los mejores resultados obtenidos de las 110 posibles combinaciones tanto para MFCC_0 como para PLP_0 para la clasificación de oclusivas y nasales respectivamente:

C	γ	Características	Tasa de acierto (%)
2^{3}	2^{-15}	MFCC_0	88.81
2^{-1}	2^{-13}	MFCC_0	**88.93**
2	2^{-3}	PLP_0	88.27
2	2^{-5}	PLP_0	88.13

Tabla 6. Resultados de la clasificación de oclusivas, con 5 % de los datos de train y test.

C	γ	Características	Tasa de acierto (%)
2^{3}	2^{-15}	MFCC_0	**91.67**
2^{-1}	2^{-13}	MFCC_0	91.50
2	2^{-3}	PLP_0	91.36
2	2^{-5}	PLP_0	91.36

Tabla 7. Resultados de la clasificación de nasales, con 5 % de los datos de train y test.

Como se puede observar en ambas tablas las tasas de acierto de los mejores clasificadores no son muy sensibles a los parámetros. Tanto para oclusivas como para nasales la mejor tasa de acierto la obtenemos con los coeficientes MFCC_0. La tasa de acierto máxima que conseguimos para la clasificación de oclusivas es de 88.93% y para la clasificación de nasales conseguimos un 91.67%. Esta tasa la hemos conseguido solo con el 5% de los datos de test. En la siguiente tabla podemos ver la tasa de acierto conseguida con cada uno de los clasificadores elegidos, utilizando todo el conjunto de test.

C	γ	Características	Rasgo Fonético	Tasa de acierto (%)
2^{3}	2^{-15}	MFCC_0	Nasales	**91.36**
2^{-1}	2^{-13}	MFCC_0	Oclusivas	**88.41**

Tabla 8. Resultados obtenidos con todo el conjunto de test para el clasificador de nasales y oclusivas.

Se puede observar que la tasa de acierto ha variado muy poco de utilizar solo el 5 % de test a utilizar todo el conjunto de test. Podemos intuir por los resultados que el conjunto de test seleccionado del 5 % representa muy bien el total del conjunto.

3.3.3 Conclusiones

Como se comentó anteriormente los requisitos que queremos que cumplan nuestros clasificadores de rasgos fonéticos son dos: máxima tasa de acierto y bajo coste computacional.

Para el clasificador de nasales si observamos la Tabla 3 y 8, el clasificador que cumple nuestro primer requisito, máxima tasa de acierto, es el clasificador SVM con los parámetros $C=2^{3}$ y $\gamma=2^{-15}$ que consigue una tasa de acierto de 91.36 % frente al 76.20 % conseguido por la red neuronal con 18 neuronas en la capa oculta y de entrada. Pero si nos fijamos en el segundo requisito, bajo coste computacional, el clasificador SVM no lo cumple, porque como se comentó en el punto 2.2.2 los SVM no pueden trabajar con grandes bases datos como es el caso de la TIMIT, de echo sólo se ha podido utilizar 5% del total de datos de entrenamiento de TIMIT, por el contrario la red neuronal puede trabajar con toda la base de datos TIMIT y su coste computacional es mucho menor que el del SVM, por lo tanto respecto a este requisito la mejor opción sería la red neuronal.

En conclusión, vamos a elegir la red neuronal como clasificador de nasales debido sobre todo a su bajo coste computacional y a que la tasa conseguida de acierto es bastante buena.

En el clasificador de oclusivas vuelve a ocurrir lo mismo, aunque en este caso la diferencia entre tasas máximas conseguidas por los distintos clasificadores no es tan grande como en el caso de las nasales, conseguimos una tasa de acierto de 88.41 % con el SVM y una tasa de acierto de 77.70 % con la red neuronal. Por lo que nuestra elección vuelve a ser la red neuronal para desarrollar el clasificador.

Por último cabe destacar que la parametrización de la señal de voz con la que funciona mejor la red neuronal y la máquina de soporte vectorial es distinta, puesto que la red neuronal consigue las mejores tasas de acierto con los coeficientes PLP_0 y la máquina de soporte vectorial con los coeficientes MFCC_0.

En conclusión, en la siguiente fase del proyecto, donde se realizarán los clasificadores de rasgos fonéticos para la base de datos Wall Street Journal (WSJ0) utilizaremos las redes neuronales.

4. Experimentos de reconocimiento de voz de gran vocabulario

En los puntos previos se ha explicado la teoría y el funcionamiento de diferentes clasificadores, llegando a la conclusión que los clasificadores de rasgos fonéticos que se van a implementar en nuestros experimentos finales son las redes neuronales.

A partir de ahora se utilizará como base de datos la Wall Street Journal (WSJ0), que va a ser explicada en el punto 4.1. Después se comentará como se han implementado los clasificadores para esta base de datos, punto 4.2 y por último en el punto 4.3 se utilizarán las salidas de estos clasificadores junto con las salidas de la extracción de características para formar un nuevo vector de características que será el que introduzcamos al reconocedor, para realizar los últimos experimentos que ya fueron explicados brevemente en el punto 2.3.

4.1 Wall Street Journal (WSJ0)

WSJ0 o CSR-I (WSJ0) [23] es un corpus específico de voz continua en inglés desarrollado por ARPA continuous Speech Recognition que permite el desarrollo de experimentos en diferentes entornos acústicos.

El conjunto de entrenamiento es el estándar SI-84 que consiste en 7134 frases grabadas desde 84 hablantes, y el conjunto de test es el estándar Nov'92 no verbalizado 5K vocabulario cerrado (wsj-5k) que consiste en 330 frases grabadas de 8 hablantes. La señal de voz fue muestreada a 16 KHz.

El nombre de la base de datos es Wall Street Journal porque las frases grabadas corresponden con noticas del periódico Wall Street Journal que han sido leídas.

4.2 Clasificadores de rasgos fonéticos para WSJ0

Para construir los clasificadores vamos a utilizar redes neuronales. Los rasgos fonéticos que vamos a clasificar son los explicados en el punto 3.1, que son: vocal, nasal, fricativa, africada, líquida, oclusiva y también se realizará un clasificador de silencio. La elección de clasificar estos rasgos y no otros, es porque estos grupos se corresponden con los principales grupos de rasgos fonéticos y además agrupan todos los fonemas existentes en el inglés.

El primer paso para construir los clasificadores es obtener sus entradas, y como ya se dijo anteriormente tienen que estar etiquetadas, es decir tenemos que buscar en los datos que tramas corresponden con vocales, fricativas, oclusivas, etc. Para realizar el etiquetado se utiliza la siguiente información:

- En HTK se define un fichero que contiene todos los grupos posibles de rasgos fonéticos que hay en el inglés, de aquí obtenemos la información de que fonemas pertenecen a cada uno de los grupos.
- Un fichero de alineamiento, que nos dice por qué fonemas está formada cada frase y además dónde empieza y acaba cada fonema en el tiempo. Este fichero se obtiene mediante un alineamiento forzado con un modelo, en un paso previo.

Los datos que utilizamos para buscar vocales, nasales,… son los vectores de características de 39 dimensiones del conjunto de entrenamiento de WSJ0 que se han obtenido en la fase de extracción de características del reconocedor. La parametrización que se ha utilizado es MFCC_0_D_A_Z, es decir, los 39 coeficientes provienen de: los 13 primeros coeficientes estáticos (MFCC) incluido el coeficiente cero (0), además estos trece coeficientes tiene media cero (Z), 13 coeficientes delta (D) y 13 coeficientes de aceleración (A).

Pero para que los clasificadores trabajen mejor, se les introduce en lugar de datos de 39 dimensiones de 65, que vienen de: los 13 coeficientes estáticos MFCC de la trama que nos interesa y el resto son los 13 coeficientes estáticos de las dos tramas posteriores y anteriores, en lugar de los coeficientes delta y de aceleración.

Lo último que falta para poder entrenar ya a los clasificadores es elegir el número de neuronas de sus capas, el de la capa de salida ya se sabe que es dos porque se quiere realizar clasificadores binarios. Para las otras dos capas el número de neuronas es el mismo, y tiene que ser mayor o igual que 14 visto los resultados que se obtuvieron en los experimentos del punto 3.3. Tras varias pruebas con una parte de los datos de entrenamiento se decidió que todos los clasificadores tendrían 21 neuronas en sus capas de entrada y oculta excepto el clasificador de nasales que con 20 neuronas obtienen buenos resultados. El tiempo empleado en el entrenamiento de las redes fue desde una semana para el clasificador de africadas, ya que el número de ejemplos no era muy grande a casi 4 semanas para el clasificador de vocales y fricativas, pues estamos hablando de cientos de miles de ejemplos.

En la Tabla 9 se muestran los clasificadores realizados con sus tasas de acierto en el entrenamiento.

Rasgo Fonético	Neuronas C.entrada	Neuronas C.oculta	Tasa de Acierto (%)
Fricativa	21	21	92.93
Vocal	21	21	93.06
Nasal	20	20	90.93
Liquida	21	21	92.58
Africada	21	21	88.34
Oclusiva	21	21	90.60
Silencio	21	21	87.84

Tabla 9. Tasa de acierto en el entrenamiento de los clasificadores.

Una vez entrenadas las redes, el siguiente paso consiste en concatenar las salidas de los clasificadores con los vectores de características. Para cada vector obtenemos sus salidas correspondientes de cada clasificador, y las concatenamos a él, es decir que vamos a tener un vector de características de más de 39 dimensiones. Este último paso se realiza tanto para los datos de entrenamiento como para los de test, puesto que van a ser estos datos los que se van a utilizar en la última fase con el reconocedor.

4.3 Experimentos con el reconocedor

Los últimos experimentos del proyecto se van a realizar con el reconocedor HTK, y con una modificación de este, que se llama HTK_STUZ realizado por el grupo GTC del I3A.

Reconocedor de referencia

Comenzaremos explicando el sistema desde el cual vamos a partir, cuya tasa de error será nuestro resultado de referencia (baseline). A partir de este sistema, que es un sistema de reconocimiento automático del habla (RAH) estándar, Figura 25, se irán incluyendo nuevos bloques en el reconocedor para intentar mejorar la tasa de error.

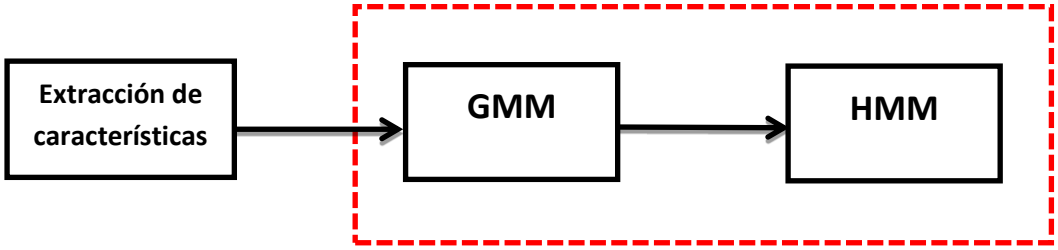

Figura 25. Sistema base. Reconocedor WSJ0.

Las entradas al sistema de la Figura 25 son los datos de WSJ0. Al pasar por el primer bloque obtenemos sus vectores de características de 39 dimensiones con parametrización MFCC_0_D_A_Z, que son las entradas al reconocedor.

El reconocedor consta de dos fases: front-end, donde se realiza la extracción de características; y back-end que se divide en dos etapas. En la primera etapa se realizan los modelos de los trifonemas, cada modelo es un HMM (Modelo Oculto de Markov) y cada estado lo vamos a modelar con una mezcla de 8 Gaussianas, excepto el modelo del silencio que se realiza con 16 Gaussianas (bloque GMM). Para realizar este modelado se utilizan solo los datos de entrenamiento de WSJ0. La segunda etapa, consiste en la decodificación de los HMM, es decir, se cogen los datos de test y se reconocen, el resultado que obtenemos es la tasa de error del reconocedor (bloque HMM).

Antes de llevar a cabo las modificaciones en nuestro sistema base, se realizarán unos experimentos previos que van a consistir en realizar la extracción de características con el programa OpenSmile, en lugar de con el bloque implementado en nuestro sistema. Se van a calcular tanto coeficientes MFCC con distintas configuraciones como coeficientes PLP, todos los vectores tendrán una dimensión de 39. Una vez que se tienen calculados se van a utilizar como entradas al reconocedor, las tasas de error obtenidas se encuentran en la Tabla 10. En ella se puede ver que no se consigue mejorar la tasa de error con ninguna parametrización, y que la mejor tasa de error se consigue con la parametrización MFCC_0_D_A_Z, que es la misma que implementa nuestro bloque de extracción de características, por lo que dejaremos la parametrización de nuestro sistema base.

Características		Tasa de error (%)
Baseline	MFCC_0_D_A_Z	**8.28**
OpenSmile	MFCC_0_D_A	8.74
OpenSmile	MFCC_0_D_A_Z	8.39
OpenSmile	MFCC_E_D_A	9.34
OpenSmile	MFCC_E_D_A_Z	9.40
OpenSmile	PLP_0_D_A	9.27
OpenSmile	PLP_0_D_A_Z	8.71
OpenSmile	PLP_E_D_A	9.49
OpenSmile	PLP_E_D_A_Z	8.87

Tabla 10. Resultados obtenidos para distintos tipos de parametrizaciones de la señal de voz.

Primera modificación: introducción de los clasificadores de rasgos fonéticos

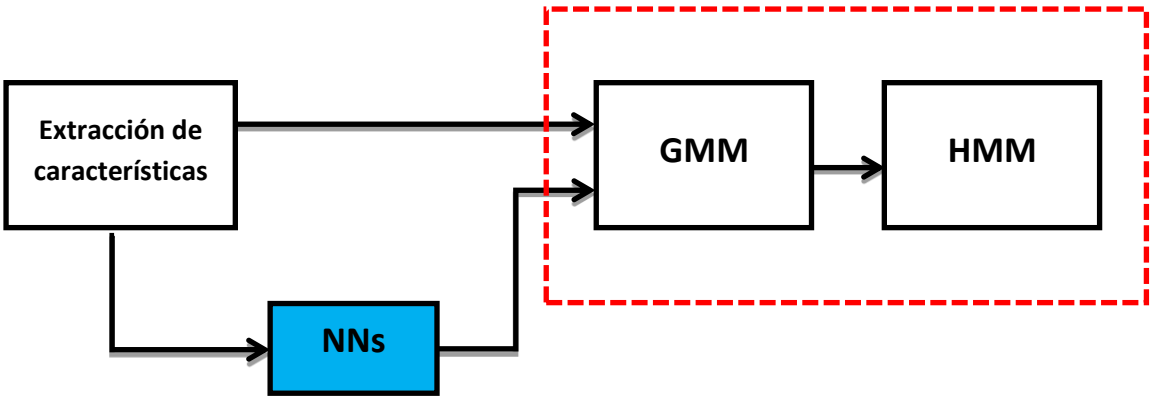

Figura 26. Reconocedor base + clasificadores de rasgos fonéticos (CRF)

En la Figura 26 se puede ver la primera modificación introducida, que consiste en que ahora al reconocedor no sólo le entran las salidas del extractor de características, sino que también tiene como entradas a las salidas de las redes neuronales que hemos implementado en el punto anterior.

En un primer momento solo vamos a utilizar 5 de los 7 clasificadores que hemos hecho, que son: el clasificador de nasales, liquidas, africadas, oclusivas y vocales. Por lo tanto las entradas al reconocedor ahora son unos vectores de características de 44 dimensiones (39 + 5), los vectores procedentes del extractor de características no sufren ninguna modificación solamente se les concatenan las salidas de los clasificadores para formar los nuevos vectores.

Si nos fijamos en la Figura 26, la parte del reconocedor no se ve modificada, lo único que tendremos que hacer es cambiar unos ficheros de configuración para que el reconocedor pueda trabajar con los datos de 44 dimensiones.

Experimento	Tasa de error (%)
Baseline	**8.28**
Reconocedor base + CRF	11.15

Tabla 11. Resultados de los dos primeros experimentos

La idea por la que se han añadido las salidas de los clasificadores a los vectores de características ya existentes es para dar más información al reconocedor y así mejorar su tasa de reconocimiento, pero como se puede observar en la Tabla 11 esto no es así sino que incluso empeoramos la tasa de error de forma considerable. Este mal resultado era de esperar, tiene su explicación en la naturaleza no gaussiana de las salidas de los clasificadores, estas salidas tienen valores entre cero y uno. El problema está en que el reconocedor tal y como está diseñado solo admite datos gaussianos, y al introducirle una parte que no lo son, no los sabe interpretar bien. En trabajos anteriores se ha propuesto la siguiente solución [12], Figura 27.

Segunda modificación: Bloque PCA

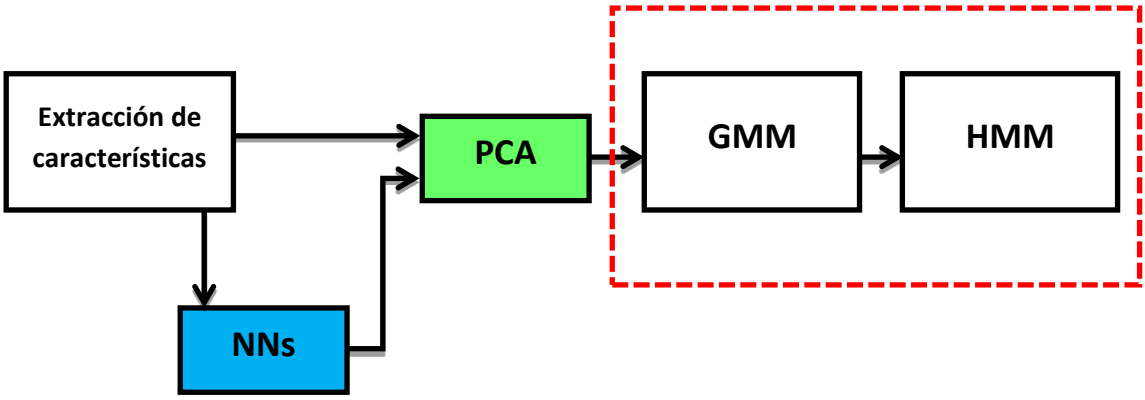

Figura 27. Reconocedor base + CRF + PCA.

El nuevo sistema que planteamos difiere del de la Figura 26 en que le hemos añadido un nuevo bloque, el cual realiza la de-correlación PCA. El bloque PCA se introduce para adaptar las salidas de los clasificadores a los modelos gaussianos del reconocedor. El objetivo de PCA es obtener las direcciones principales de variación de los datos, estimando la matriz de covarianza, de la cual calculamos sus autovalores y autovectores. La matriz A contiene estos autovectores ordenados de forma decreciente según el valor de sus autovalores, obteniendo así los vectores base para la proyección de los datos. En nuestro caso tendremos unos vectores base de 44 dimensiones que contienen la misma información que nuestros datos antes de realizarles la de-correlación PCA. Podemos retener información, es decir, utilizar estos vectores base pero con una dimensión menor de 44 y seguir manteniendo la información relevante de nuestros datos.

Experimentos	Dimensión	Tasa de error (%)
Baseline	39	**8.28**
Reconocedor base + CRF	44	11.15
Reconocedor base + CRF + PCA	44	10.59
Reconocedor base + CRF + PCA	39	9.75
Reconocedor base + CRF + PCA	21	13.69

Tabla 12. Resultados obtenidos al añadir el bloque PCA con la matriz A con distintas dimensiones.

En los resultados de la Tabla 12 podemos ver que cuando la dimensión de la matriz A solo es de 21, la tasa de error empeora considerablemente. Esto es debido a que se ha eliminado información importante al truncar la matriz de 44 a 21 dimensiones. Respecto a los otros dos casos, cuando A tiene la misma dimensión que nuestros datos conseguimos una mejor tasa que si no aplicamos PCA; y cuando A tiene una dimensión de 39 se obtiene una tasa de 9.75%, la mejor, pero sigue estando por encima del 8.28%.

Por lo tanto al añadir el bloque PCA hemos mejorado pero todavía estamos lejos de nuestro objetivo. Por lo que se va a plantear otra modificación pero esta vez dentro del propio reconocedor. A partir de ahora utilizaremos el reconocedor HTK_STUZ.

Tercera modificación: Introducción de nuevos tipos de modelos en la fase de reconocimiento

Como ya se explicó en el punto 2.3 esta tercera modificación supone una novedad en el estado del arte de los sistemas RAH de gran vocabulario. Se van a plantear tres sistemas posibles, cuya diferencia entre ellos es el nuevo modelo al que se van a adaptar las salidas de las redes neuronales. Ya hemos visto que utilizando el bloque PCA conseguimos que las salidas de las redes se adapten mejor a los modelos gaussianos que utiliza el reconocedor, pero ahora lo que queremos es utilizar otros tipos modelos a los que si se adapten las salidas de las redes sin tener que realizar ningún preprocesado.

Se plantean tres modelos: distribución Beta, redes Bayesianas y distribución de Bernoulli. La distribución Beta trabaja con datos de entrada que se encuentran en el rango (0,1), y las otras dos trabajan con datos discretos {0,1}, por lo que los tres pueden trabajar con las salidas de los clasificadores, en el caso de las redes Bayesianas y de la Bernoulli habrá que cuantificar las observaciones. En las Figuras 28, 29 y 30 se pueden ver los tres sistemas que se plantean.

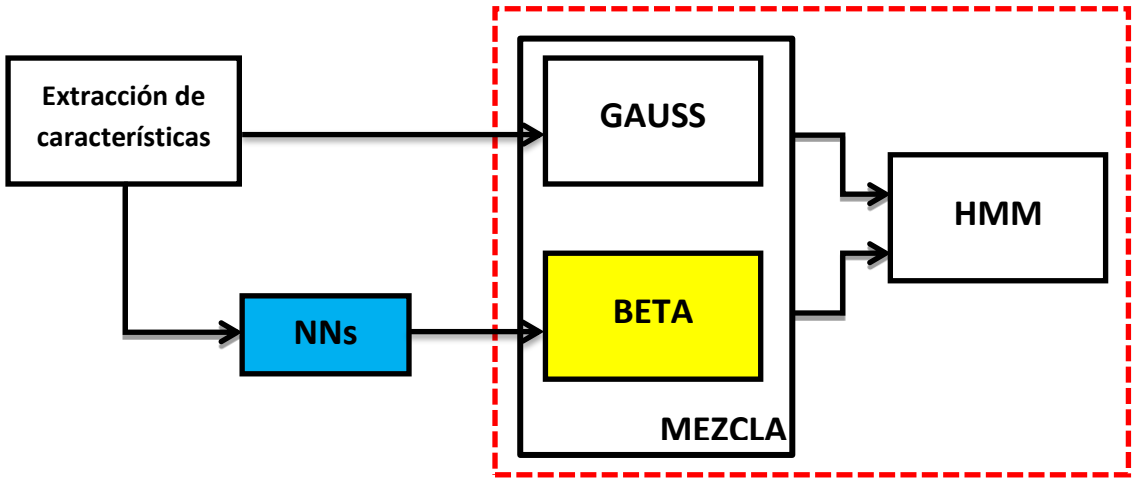

Figura 28. Sistema RAH que añade a su fase de reconocimiento la distribución Beta.

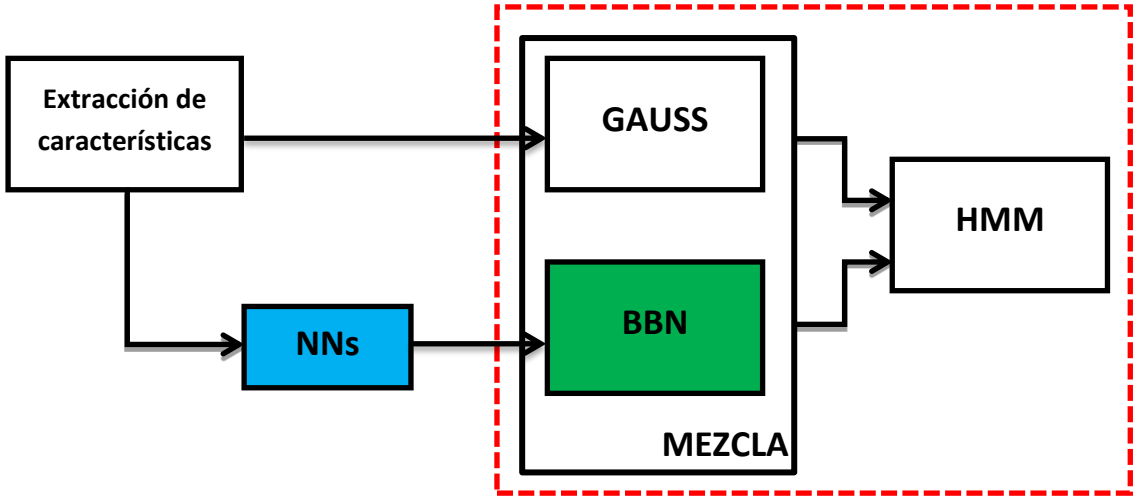

Figura 29. Sistema RAH que añade a su fase de reconocimiento las redes Bayesianas (BBN)

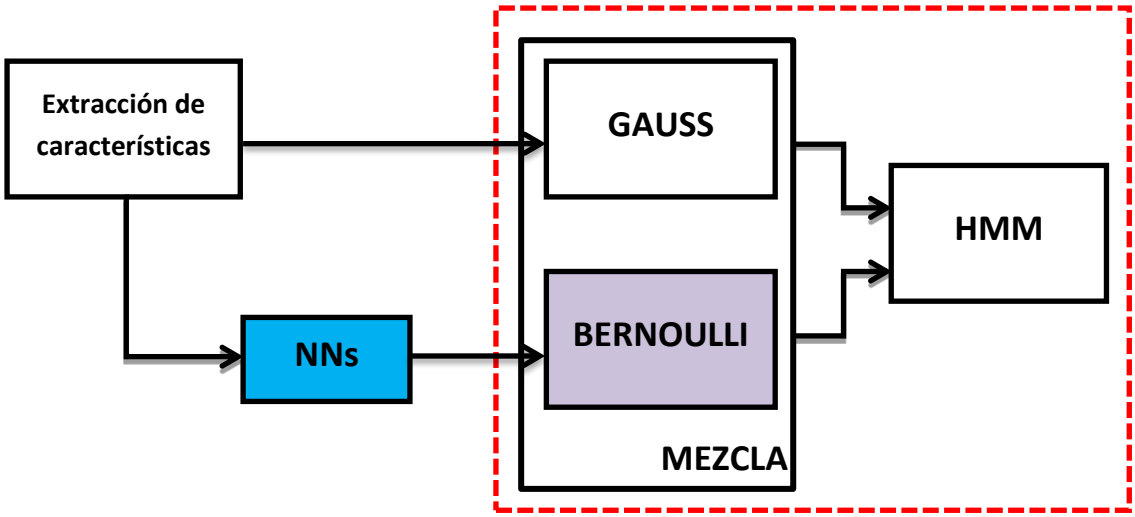

Figura 30. Sistema RAH que añade a su fase de reconocimiento la distribución Bernoulli.

Por lo tanto en estos sistemas los vectores de características son de 44 dimensiones (39 +5), las 39 primeras dimensiones serán las entradas a los modelos de Gaussianas y las otras 5 dimensiones serán las entradas a los otros modelos. En la Tabla 13 tenemos los primeros resultados con estos nuevos sistemas.

Experimentos	Tasa de error (%)
Baseline	**8.28**
Reconocedor GMM + Beta	8.33
Reconocedor GMM + Bernoulli	8.33
Reconocedor GMM + BBN	**8.28**

Tabla 13. Primeros resultados combinando modelos gaussianos con otros tipos de modelos en la fase de reconocimiento.

Los primeros resultados que obtenemos son bastante alentadores, pues como se puede ver en la Tabla 13 si utilizamos redes Bayesianas y mezclas de Gaussianas en el reconocedor obtenemos la misma tasa de error que nuestro sistema base, y en los otros dos casos conseguimos un 8.33%.

Obviamente nuestro objetivo de mejorar el 8.28% aún no se ha conseguido pero no se está lejos de hacerlo. Para conseguirlo tenemos que encontrar qué puede estar pasando para que no se obtengan tasas de error más bajas. Lo primero que nos planteamos es dar pesos a los dos tipos de modelos, es decir, cuando reconocemos tenemos dos tipos de modelos: Gaussianos y Beta, redes Bayesianas o Bernoulli, pero no tienen por qué tener la misma importancia o el mismo peso a la hora de reconocer, por lo que vamos a darles diferentes pesos [22]. En el Anexo VI se explica la manera de aplicar estos pesos. Estos experimentos los vamos a realizar solo con las distribuciones Beta y Bernoulli porque las redes Bayesianas ya nos dan una tasa de 8.28%, además la Bernoulli la podemos ver como una red Bayesiana de orden

cero, y por último el tiempo que utilizan las redes Bayesianas para reconocer es el doble que el utilizado por los otros dos modelos. En la Tabla 14 están los resultados obtenidos.

Experimentos	Tasa de error (%)
Baseline	**8.28**
Reconocedor GMM + Beta	8.33
Reconocedor GMM + Beta peso1	8.89
Reconocedor GMM + Beta peso2	8.52
Reconocedor GMM + Bernoulli	8.33
Reconocedor GMM + Bernoulli peso1	8.39
Reconocedor GMM + Bernoulli peso3	8.41

Tabla 14. Tasas de error obtenidas al aplicar distinto peso a las distribuciones. En todos los casos el peso asignado a las mezclas de Gaussianas (GMM) es de 1, y en el resto de los casos sus valores son: peso1=2, peso2=1.5 y peso3=2.5.

Se han realizado más experimentos a parte de los que se han expuesto en la Tabla 14, pero solo se mencionan aquellos que nos han dado mejor resultado. Se sigue sin conseguir mejorar las tasas, por lo que se tendrá que pensar otra solución.

Otra opción es fijarnos en nuestros clasificadores de rasgos fonéticos que son las entradas a los nuevos modelos, ya que pueden no estar funcionando correctamente. Por lo que se van a realizar los siguientes experimentos con la distribución Bernoulli que es el sistema que más rápido reconoce. Vamos a reconocer, pero eliminando en cada experimento la salida de un clasificador, para observar que ocurre si esa información no está. Para este experimento se han utilizado los pesos, es decir la GMM tienen peso igual a uno y la Bernoulli tiene un peso de dos, esto es así porque al realizar los experimentos de eliminar los clasificadores hemos obtenido mejores tasas cuando asignábamos estos pesos a las distribuciones.

Experimentos	Tasa de error (%)
Baseline	8.28
Reconocedor GMM + Bernoulli	8.39
Reconocedor GMM + Bernoulli Sin nasales	8.37
Reconocedor GMM + Bernoulli Sin liquidas	10.37
Reconocedor GMM + Bernoulli Sin africadas	8.85
Reconocedor GMM + Bernoulli Sin oclusivas	8.56
Reconocedor GMM + Bernoulli Sin vocales	**8.26**

Tabla 15. Tasas de error obtenidas eliminando en cada experimento la salida de un clasificador de rasgo fonético. En todos los casos la GMM tiene un peso=1 y la Bernoulli un peso=2.

Como podemos ver en la Tabla 15 se ha conseguido mejorar la tasa de error en un 8.26% al eliminar la salida del clasificador de vocales. Se puede observar que si eliminamos los demás clasificadores la tasa aumenta, excepto en el caso de las nasales, lo que se puede deducir de aquí es que el clasificador de vocales no ayuda en el reconocimiento e incluso se puede decir que puede que introduzca ruido ya que hace que empeore considerablemente la tasa. Se va a comprobar si para la distribución Beta y las redes Bayesianas se da el mismo comportamiento con el clasificador de vocales.

Experimentos	Tasa de error (%)
Baseline	8.28
Reconocedor GMM + Beta Sin vocales	8.26
Reconocedor GMM + BBN Sin vocales	**8.20**
Reconocedor GMM + Bernoulli Sin vocales	8.26

Tabla 16. Tasas de error conseguidas para los tres sistemas propuestos eliminando la salida del clasificador de vocales.

Por los resultados obtenidos en los tres sistemas, el clasificador de vocales estaba empeorando la tasa, esto sucede porque el clasificador introduce falsas alarmas, es decir, clasifica vocales donde no las hay. Por lo que vamos a quitar su salida del vector de características, ahora se tendrán datos de 43 dimensiones. Además hemos conseguido mejorar la tasa, consiguiendo un 8.20% al utilizar las redes Bayesianas, por lo que nuestro objetivo ya estaría cumplido.

Mirando en Matlab el comportamiento de las salidas de nuestros clasificadores, observamos que la mayoría de ellas no tienen valores superiores a 0.4, y entonces nos plantemos que igual se puede utilizar este dato para mejorar aún más la tasa de error. Tanto las redes Bayesianas como la Bernoulli necesitan un umbral, ya que solo aceptan datos cuyo valor sea cero o uno. En un principio se había puesto que este umbral valiera 0.5, es decir, que los datos que estuvieran por debajo de ese valor pasaban a valer 0 y los que estaban por encima 1. Si la mayoría de nuestros datos no superaba ese umbral lo que se estaba haciendo es eliminar información que podría ser útil, por eso nos plantemos calcular la tasa de error que obtendríamos con el sistema que utiliza la distribución Bernoulli para diferentes valores de umbral. En todos los casos utilizamos el vector de características de 43 dimensiones (sin clasificador de vocales) y a la distribución Gaussiana le asignamos un peso = 1 y a la Bernoulli un peso = 2.

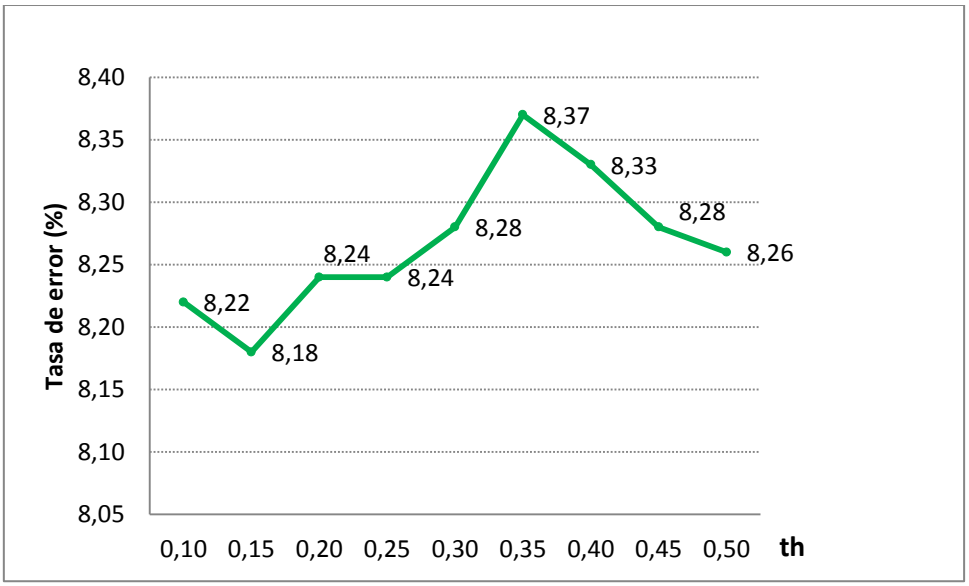

Figura 31. Tasa de error en función del valor umbral en la distribución Bernoulli.

Se consigue una tasa de error de 8.18% si el valor del umbral es 0.15, por lo que hemos conseguido mejorar la tasa de 8.20% que se obtenía con las redes Bayesianas. Como se puede ver en la Figura 31 la elección del valor del umbral sí que afecta a la tasa de error.

Como últimos experimentos a llevar a cabo en el presente proyecto se van a introducir las salidas de los otros dos clasificadores. En el punto 4.2 se desarrollaron 7 clasificadores de los cuales solo se han utilizado 5 porque los otros dos, el clasificador de fricativas y el de silencio, no habían terminado de entrenarse. Por lo tanto los nuevos vectores de entrada tienen una dimensión de 45 (39 +6), seguimos sin utilizar el clasificador de vocales puesto que ya se ha visto que funciona todo mejor sin utilizarlo. En la Tabla 17 podemos ver las tasas de error obtenidas:

Experimentos	Dimensión	Tasa de error (%)
Baseline	39	8.28
Reconocedor GMM + Beta	45	8.84
Reconocedor GMM + BBN	45	**8.07**
Reconocedor GMM + Bernoulli	45	8.16

Tabla 17. Resultados obtenidos añadiendo el clasificador de fricativas y el de silencio.

Al introducir estos dos nuevos clasificadores se ha conseguido una tasa del 8.07 % con las redes Bayesianas, la mejor tasa que se ha conseguido en todos los experimentos. Por lo que parece ser que las salidas de estos clasificadores si ayudan al reconocedor, aunque en el caso de la distribución Beta ocurre lo contrario, la hemos empeorado. La razón por la que las redes Bayesianas [26] obtienen una mejor tasa que la distribución de Bernoulli, que como ya hemos dicho antes se puede ver como una red Bayesiana de orden cero, es porque las redes Bayesianas capturan las dependencias que hay entre las distintas dimensiones y la Bernoulli trata cada dimensión como independiente.

5. Conclusiones y líneas futuras

En el presente proyecto se han implementado una serie de clasificadores de rasgos fonéticos para intentar mejorar la tasa de error del reconocedor HTK para la base de datos Wall Street Journal (WSJ0). Lo que se ha hecho es añadir las salidas de estos clasificadores a los vectores de características comunes para introducir así más información al reconocedor. Se partió de una tasa de error del 8.28% que conseguía el reconocedor HTK con una configuración estándar, por lo que el objetivo era superarla.

La novedad que se incluye en este proyecto es que las salidas de estos clasificadores de rasgos fonéticos van a ser las entradas de unos nuevos modelos que se implementan en el reconocedor y que no son gaussianos, ya que los modelos Gaussianos implementados en el HTK para crear los modelos de trifonemas no funcionan bien con las salidas de los clasificadores, puesto que sus valores están entre cero y uno. Los modelos que se presentan para implementar en el reconocedor son tres: distribución Beta, distribución de Bernoulli y redes Bayesianas. Con los tres modelos se ha conseguido reducir la tasa del 8.28%. Utilizando las redes Bayesianas se ha conseguido la mejor tasa, de un 8.07%, las entradas al reconocedor estaban formadas por los vectores de 39 dimensiones comúnmente utilizados, concatenados con las salidas de 6 clasificadores de rasgos fonéticos: nasales, liquidas, africadas, oclusivas, fricativas y silencio.

A partir de los experimentos realizados con esta nueva manera de reconocer los datos de voz, se abre la puerta a numerosos experimentos que podrían reducir aún más la tasa de error del reconocedor, como por ejemplo:

- En el presente proyecto sólo se han realizado seis clasificadores de rasgos fonéticos y un clasificador de silencio, una línea de investigación sería clasificar más rasgos fonéticos, e incluso llegar a hacer clasificadores para un solo fonema específico. En este último caso, se tendría que mirar que fonemas se confunden más en el reconocimiento, y esos fonemas serían los que tendrían su propio clasificador.
- Utilizar como clasificadores de rasgos fonéticos las máquinas de soporte vectorial (SVM), en este proyecto se ha observado que la tasa de acierto de clasificación es mejor que la obtenida por las redes neuronales, pero para llevarlo a cabo se tendría que investigar la manera de extraer de toda la base de datos aquellos que sean los más representativos, ya que los SVM no pueden trabajar con gran cantidad de datos.
- Una línea de investigación que ya se está llevando a cabo es la de utilizar las características bottle-neck, consiste básicamente en utilizar redes neuronales como

clasificadores de rasgos fonéticos pero en lugar de coger sus salidas como entradas al reconocedor coger las salidas de la capa intermedia.

- Debido al tiempo que lleva entrenar a los clasificadores, tuvimos que elegir el tipo de parametrización a usar, sin poder probar con más de un tipo. Por ello otra línea de investigación sería la de utilizar otro tipo de parametrizaciones para realizar los clasificadores, como por ejemplo los coeficientes PLP o los coeficientes RASTA, que fueron nombrados en el Capítulo 2. Se podrían obtener buenos resultados visto los experimentos realizados en el Capítulo 3, donde los coeficientes PLP conseguían mejor tasa de acierto en algunos clasificadores.

6. Referencias

[1] Huang, X., Acero, A., & Hon, H. W., others. (2001). *Spoken language processing*. *Processing*. Prentice Hall PTR New Jersey. Retrieved March 29, 2011, from http://www.library.wisc.edu/selectedtocs/bd025.pdf.

[2] Ellis, D. (2000). Tandem acoustic modeling: Neural nets for mainstream ASR ? *Science* (pp. 1-11).

[3] Solera-Urena, R., Padrell-Sendra, J., Martin-Iglesias, D., Gallardo-Antol\'\in, A., Peláez-Moreno, C., & Diaz-de-Maria, F. (2007). SVMs for automatic speech recognition: a survey. *Progress in nonlinear speech processing*, 190–216. Springer. Retrieved March 29, 2011, from http://www.springerlink.com/index/R828226517290181.pdf.

[4] Grézl, F., Karafiát, M., Kontár, S., & Cernocky, J. (2007). Probabilistic and bottle-neck features for LVCSR of meetings. *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* (Vol. 4, p. IV–757). IEEE. Retrieved March 30, 2011, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4218211.

[5] Hermansky, H. (1990). Perceptual liner predictive (PLP) analysis of speech (Vol. 87, pp. 1738-1752).

[6] Hermansky, H., Morgan, N., Bayya, A., Kohn, P., & Drive, D. (1992). Rasta-plp speech analysis technique. *Management*, 121-124.

[7] Sakoe, H., Isotani, R., Yoshida, K., Iso, K.-I., & Watanabe, T. (1989). Speaker-independent word recognition using dynamic programming neural networks. *International Conference on Acoustics, Speech, and Signal Processing* (pp. 29-32). Ieee. doi: 10.1109/ICASSP.1989.266355.

[8] Iso, K.-I., & Watanabe, T. (1990). Speaker-independent word recognition using a neural prediction model. *International Conference on Acoustics, Speech, and Signal Processing* (pp. 441-444). Ieee. doi: 10.1109/ICASSP.1990.115744.

[9] Sugiyama, M., Sawai, H., & Waibel, a H. (n.d.). Review of TDNN (time delay neural network) architectures for speech recognition. *1991., IEEE International Sympoisum on Circuits and Systems*, 582-585. Ieee. doi: 10.1109/ISCAS.1991.176402.

[10] Lee, T., & Ching, P. (1995). Recurrent neural networks for speech modeling and speech recognition. *icassp*, 3319-3322. doi: 10.1109/TNN.2011.2109735.

[11] Tang, X. (2009). Hybrid Hidden Markov Model and Artificial Neural Network for Automatic Speech Recognition. *2009 Pacific-Asia Conference on Circuits, Communications and Systems* (pp. 682-685). Ieee. doi: 10.1109/PACCS.2009.138.

[12] Hermansky, H., Ellis, D. P. W., & Sharma, S. (n.d.). Tandem connectionist feature extraction for conventional HMM systems. *2000 IEEE International Conference on*

Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100), 1635-1638. Ieee. doi: 10.1109/ICASSP.2000.862024.

[13] Ellis, D. P. W., Singh, R., & Sivadas, S. (2001). Tandem acoustic modeling in large-vocabulary recognition. *icassp* (p. 517–520). IEEE. Retrieved March 30, 2011, from http://www.computer.org/portal/web/csdl/doi/10.1109/ICASSP.2001.940881.

[14] Gong, Y. (1995). Speech recognition in noisy environments: A survey. *Speech communication*, *16*(3), 261–291. Elsevier. Retrieved March 30, 2011, from http://linkinghub.elsevier.com/retrieve/pii/016763939400059J.

[15] Ghaemmaghami, M. P., Sametit, H., Razzazi, F., & Babaali, B. (2009). Robust Speech Recognition Using MLP Neural Network in Log-Spectral Domain. *Strategies*, 467-472.

[16] Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine* (Vol. 4, pp. 4-22). doi: 10.1109/MASSP.1987.1165576.

[17] Pérez-cruz, F. (2004). Kernel methods and their potential use in signal processing. An overview and guidelines for future development. *IEEE Signal Processing Magazine*, (May).

[18] Hsu, C. W., Chang, C. C., & Lin, C. J., others. (2003). A practical guide to support vector classification. *Bioinformatics*.

[19] Eyben, F., Woellmer, M., & Schuller, B. (2010). the Munich open Speech and Music Interpretation by Large Space Extraction toolkit.

[20] Yasser EL-Manzalawy and Vasant Honavar, WLSVM: Integrating LibSVM into Weka Environment, 2005. Software available at http://www.cs.iastate.edu/~yasser/wlsvm

[21] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallet, and N. L.Dahlgren, \The DARPA TIMIT acoustic-phonetic continuous speech corpusCDROM," National Institute of Standards and Technology, Gaithersburg, MD, Tech.Rep., 1993.

[22] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu,G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. The HTK Book. Cambridge University Engineering Department,Cambridge University Engineering Department, Cambridgeshire, UK,for htk version 3.4 edition, December 2006.

[23] Paul, D. B., & Baker, J. M. (1992). The design for the wall street journal-based CSR corpus. *Proceedings of the workshop on Speech and Natural Language - HLT '91*, 357. Morristown, NJ, USA: Association for Computational Linguistics. doi: 10.3115/1075527.1075614.

[24] An-na, W., Yue, Z., Yun-tao, H., & Yun-lu, L. (2010). A novel construction of SVM compound kernel function. *Logistics Systems and Intelligent Management, 2010 International Conference on* (Vol. 3, p. 1462–1465). IEEE. doi: 10.1109/ICLSIM.2010.5461210.

[25] P. Murphy, D. Aha, and G. Robinson. Uci repository of machine learning databases: Machine-readable data repository. Technical report, Irvine, CA: University of California, Department of Information and Computer Science, http://www.ics.uci.edu/ mlearn/MLRepository. html, 1992.

[26] Miguel A, Ortega A, Buera L, Lleida E. Bayesian Networks for Discrete Observation Distributions in Speech Recognition. IEEE Transactions on Audio, Speech, and Language Processing. 2011;19(6):1476-1489.

[27] Jolliffe, I. T. (n.d.). Principal Component Analysis, Second Edition.

Anexo I. Extracción de características

Para llevar a cabo la extracción de características primero se realiza la adquisición de la señal de voz. La señal de voz es una señal analógica que se captura a través de un micrófono, se tiene que digitalizar, para ello se necesita un buffer típicamente de 4 a 64 KB con una frecuencia de muestreo de 16kHz y una precisión de 16 bit A/D [1]. En la práctica la frecuencia de 16kHz es suficiente puesto que el ancho de banda de la señal de voz queda bien representado con 8kHz. Una vez digitalizada la señal se pueden obtener sus características.

En este proyecto se utilizan las siguientes parametrizaciones: los coeficientes MFCC; los coeficientes PLP y por último los coeficientes RASTA. Las tres técnicas utilizan el análisis localizado de la señal de voz, que consiste en enventanar la señal con una ventana normalmente de 25ms con un desplazamiento de 10ms, tratando estos fragmentos de la señal de manera independiente.

A continuación se pasará a explicar cada técnica en detalle.

Mel-Frecuency Cepstrum Coefficients (MFCC)

El proceso de extracción de características MFCC se puede observar en la siguiente figura:

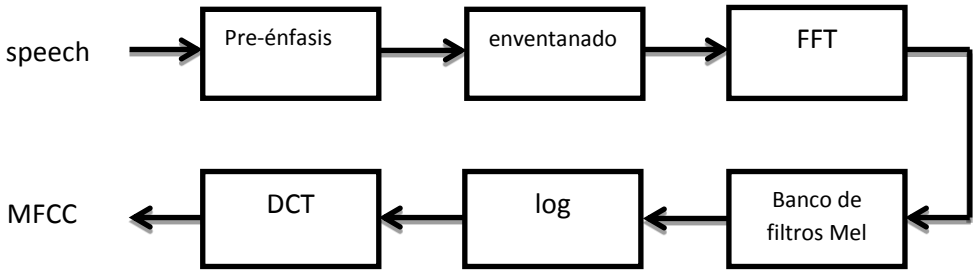

Figura I.1. Esquema del proceso de extracción de características

Sea la señal de voz x(n), pasamos esta señal por el filtro de preénfasis con K= 0.97:

$$x_p(n) = x(n) + Kx(n-1) \qquad (I.1)$$

Después a la señal se le aplica una ventana de Hamming de 25ms de ancho:

$$x_w(n) = x_p(n) * w_h(n) \qquad (I.2)$$

Una vez la señal ha sido enventanada aplicamos la FFT para obtener el espectro:

$$X_w(k) = \sum_{n=0}^{N-1} x_w(n) exp\left\{-\frac{2\pi kn}{N}\right\} \qquad (I.3)$$

En el siguiente paso, vamos a aplicar un banco de M filtros a la potencia del espectro. El banco de filtros está formado por M filtros triangulares:

$$H_m[k] = \begin{cases} 0 & k < f[m-1] \\ \dfrac{2(k - f[m-1])}{(f[m+1] - f[m-1])(f[m] - f[m-1])} & f[m-1] \le k \le f[m] \\ \dfrac{2(f[m+1] - k)}{(f[m+1] - f[m-1])(f[m+1] - f[m])} & f[m] \le k \le f[m+1] \\ 0 & k > f[m+1] \end{cases} \qquad (I.4)$$

Cada filtro calcula la media del espectro alrededor de la frecuencia central, los filtros tienen distintos anchos de banda a mayor frecuencia mayor ancho de banda, como se puede ver en la figura siguiente:

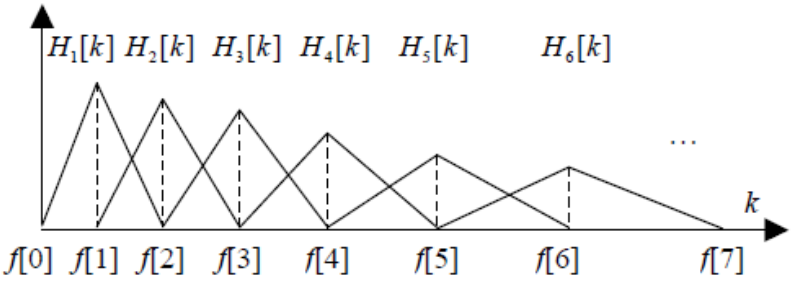

Figura I.2. Filtros triangulares para calcular el Mel-cepstrum.

Definimos las frecuencias de corte del banco de filtros como f_l y f_h en Hz, F_s es la frecuencia de muestreo en Hz, M es el número de filtros y N el número de muestras de la FFT. Los puntos f(m) están espaciados de forma no uniforme en la escala Mel:

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_l) + m\frac{B(f_h) - B(f_l)}{M+1}\right) \qquad (I.5)$$

Donde B es la escala Mel definida en [1].

Una vez definido todo esto, calculamos la log-energía de la salida de cada filtro:

$$S[m] = ln\left[\sum_{k=0}^{N-1} |X_w[k]|^2 H_m[K]\right], \qquad 0 \le m \le M \qquad (I.6)$$

Por lo tanto los coeficientes mel-cepstrum se obtienen de aplicar la DCT (Discrete Cosine Transform) a las salidas del banco de filtros:

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos\left(\frac{\pi n\left(m+\frac{1}{2}\right)}{M}\right) \qquad 0 \leq n \leq M \qquad (I.7)$$

El valor de M va de 24 a 40 según sea la aplicación. En el caso de reconocimiento de voz se usan típicamente los 13 primeros coeficientes.

Nos interesa captar los cambios temporales que se dan en el espectro ya que juegan un papel muy importante en la percepción humana y en la coarticulación. Por ello se utilizan a parte de los 13 primeros coeficientes MFCC los coeficientes delta ya que capturan esa información y los coeficientes de aceleración. Los coeficientes delta y los de aceleración no son más que la primera y segunda derivada de los coeficientes MFCC respectivamente.

En un sistema RAH normalmente se utilizan como características:

- Los 13 primeros coeficientes MFCC, coeficientes estáticos.
- 13 coeficientes delta.
- 13 coeficientes de aceleración.

Por lo tanto se suele trabajar con un vector de características de 39 dimensiones.

Predicción Lineal Perceptual (PLP)

El principio básico del análisis PLP es obtener mediante un modelo de todo polos una aproximación del espectro auditivo de la voz. En la Figura I.3 aparece el diagrama de bloques del sistema que realiza la extracción de los coeficientes PLP, a continuación pasaremos a explicar cómo se lleva a cabo este proceso.

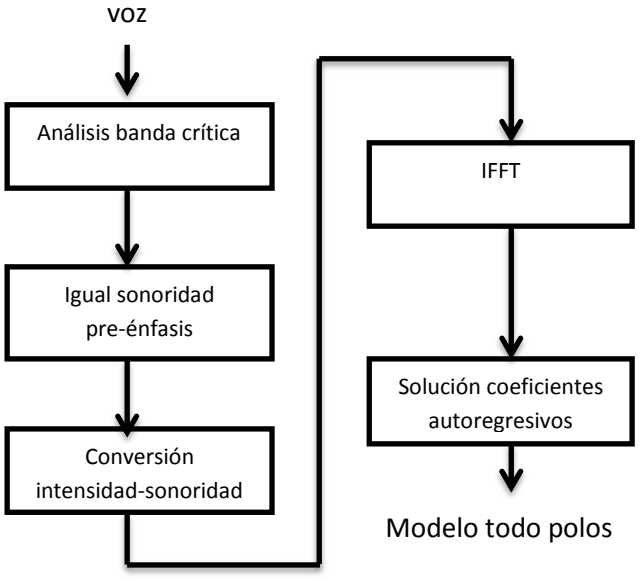

Figura I.3. Diagrama de bloques de extracción de coeficientes PLP.

Al segmento de la señal de voz se le aplica una ventana de Hamming:

$$W(n) = 0.54 + 0.46\cos[2\pi n/(N-1)] \qquad (I.8)$$

donde N es la longitud de la ventana.

La duración de la ventana suele ser de unos 20ms. Aplicamos la DFT (Transformada Discreta de Fourier) al segmento enventanado de la señal para pasarlo al dominio frecuencial. Para realizarlo se suele utilizar la FFT (Fast Fourier Transform). Después la parte real e imaginaria del espectro localizado de la señal se elevan al cuadrado y se suman para calcular el espectro localizado de potencia de la señal:

$$P(w) = Re[S(w)]^2 + Im[S(w)]^2 \qquad (I.9)$$

El espectro P(w) se desplaza de su eje frecuencial w a la frecuencia Bark Ω [5]:

$$\Omega(w) = 6ln\left\{\frac{w}{1200\pi} + [(w/1200\pi)^2 + 1]^{0.5}\right\} \qquad (I.10)$$

Donde w es la frecuencia angular en rad/s. El resultado de desplazar el espectro de potencia se convoluciona con el espectro de potencia de la curva de enmascaramiento de la banda crítica Ψ(Ω). Este paso es similar al procesado espectral en el análisis mel-cepstrum, excepto por la particular forma de la curva de la banda crítica. En la técnica PLP la forma de la curva de la banda crítica es [5]:

$$\Psi\Psi(\Omega) = \begin{cases} 0 & \Omega < -1.3 \\ 10^{2.5(\Omega+0.5)} & -1.3 \leq \Omega \leq -0.5 \\ 1 & -0.5 < \Omega < 0.5 \\ 10^{-(\Omega-0.5)} & 0.5 \leq \Omega \leq 2.5 \\ 0 & \Omega > 2.5 \end{cases} \qquad (I.11)$$

Por lo tanto nos queda:

$$\theta(\Omega_i) = \sum_{\Omega=-1.3}^{2.5} P(\Omega - \Omega_i)\Psi\Psi(\Omega) \qquad (I.12)$$

Con esta convolución conseguimos reducir la resolución espectral de Θ(Ω) con respecto al espectro original P(w). Θ(Ω) esta muestreada en intervalos de 1 Bark. El valor exacto del intervalo de muestreo se elige para que un número entero de muestras espectrales cubran toda la banda. Típicamente 18 muestras espectrales de Θ(Ω(w)) son usadas para cubrir el análisis de un ancho de banda de 0-16.9 Barks (0-5 KHz) en intervalos de 0.994 Barks.

Ahora a nuestra señal muestreada Θ(Ω(w)) le aplicamos la curva de igual-sonoridad:

$$\mathrm{E}[\Omega(w)] = E(w)\theta[\Omega(w)] \qquad (I.13)$$

El comportamiento de E(w) se aproxima a la sensibilidad del oído humano a diferentes frecuencias. Según el rango de frecuencias el filtro E(w) tiene una función de transferencia u otra.

La última operación antes de obtener el modelo de todo polos, consiste en una compresión de la amplitud del espectro:

$$\phi(\Omega) = \mathrm{E}(\Omega)^{0.33} \qquad (I.14)$$

Esta operación es una aproximación de la ley de potencia del oído, que simula la relación no lineal entre la intensidad del oído y su percepción sonora. Esta paso como el anterior los realizamos para poder obtener un modelo de todo polos de orden reducido.

El último paso es aproximar ϕ(Ω) al espectro de un modelo de todo polos usando el método de autocorrelación [1]. Los coeficientes que obtenemos del método de autocorrelación pueden ser transformados para convertirse en lo coeficientes cepstrum de un modelo de todo polos, es decir, en los coeficientes PLP.

El orden del modelo todo polos variará según el grado de detalle que queramos que tenga la aproximación del espectro auditivo de la voz.

Relative Spectral Transform (RASTA)

El análisis RASTA consiste en reemplazar el espectro localizado por un espectro estimado en el que cada canal frecuencial es filtrado paso banda con un filtro que vale cero en la frecuencia cero, este cero sirve para eliminar las variaciones lentas o los valores constantes de cada canal frecuencial y también vale cero en frecuencias superiores a las utilizadas por la dinámica de la voz, de unas decenas de Hz, para así eliminar variaciones demasiado rápidas. El nuevo espectro será menos sensible a variaciones lentas en el espectro localizado.

Los pasos a seguir en el cálculo de coeficientes RASTA-PLP son los que se muestran en la siguiente figura:

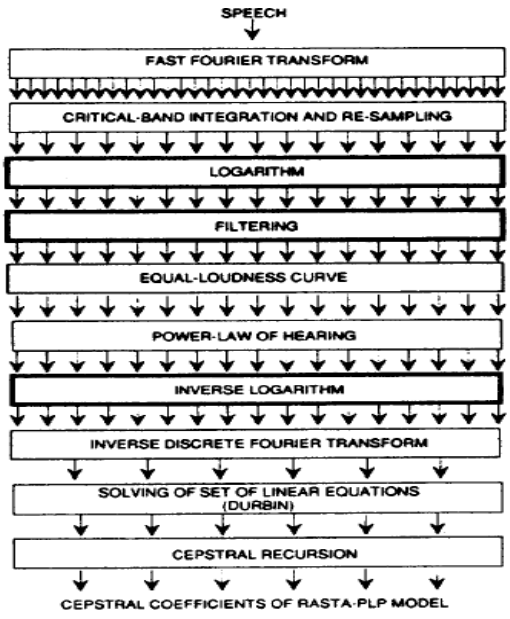

Figura I.4. Diagrama de bloques para calcular coeficientes RASTA-PLP

El análisis llevado a cabo en cada segmento de voz es:

1- Calculamos el espectro de la banda crítica (como en PLP) y calculamos su logaritmo.
2- Calculamos la derivada temporal de la señal obtenida en 1. Para calcular la derivada utilizamos una técnica de regresión lineal.

3- Realizamos un procesado no lineal, aunque este paso puede ser ignorado.
4- Pasamos la señal por un sistema IIR de primer orden.
5- Como en el sistema PLP, le aplicamos la curva de igual-sonoridad y la compresión en amplitud de 0.33 para simular la ley de potencia del oído.
6- Calculamos el logaritmo inverso de este log espectro y obtenemos así un espectro auditivo.
7- Por último, calculamos el modelo todo polos como en PLP.

Anexo II. OpenSmile

OpenSmile es un software libre que permite calcular distintos tipos de parametrizaciones para señales de voz, entre ellas, los coeficientes MFCC y PLP, los dos tipos que vamos a utilizar en nuestros experimentos. Permite distintos tipos de datos de entrada como: .wav o .raw.

Con el programa vienen unos ficheros de configuración que te permiten calcular los siguientes tipos de parametrizaciones:

- MFCC12_0_D_A_Z
- MFCC12_0_D_A
- MFCC12_E_D_A_Z
- MFCC12_E_D_A
- PLP_0_D_A_Z
- PLP_0_D_A
- PLP_E_D_A_Z
- PLP_E_D_A

La nomenclatura que utilizan es la siguiente, en primer lugar se pone el nombre de los coeficientes que se van a calcular, MFCC12 o PLP; las demás letras que se añaden se pueden poner en el orden que se quiera. Su significado es:

- 0, el coeficiente cero está incluido en los 13 coeficientes estáticos.
- E, la energía está incluida como uno de los 13 coeficientes estáticos.
- Z, los coeficientes estáticos tienen media cero.
- D, se añaden los 13 coeficientes delta
- A, se añaden los 13 coeficientes de aceleración.

Con estos ficheros de configuración se obtienen las salidas en formato htk, es decir, en un formato que lee el reconocedor HTK, por lo que nos van a ser útiles para nuestros experimentos con el reconocedor. Pero también permite obtener las salidas en otros tipos de formato, como es el caso del tipo Arff, tipo de datos que utiliza WEKA, por lo que también nos interesa por esto.

A continuación se muestra la parte del código de un fichero de configuración donde se calcula los 13 coeficientes MFCC estáticos, incluidos el coeficiente cero (0), los coeficientes delta (D) y los de aceleración (A):

"MFCC12_0_D_A.conf"

```
[melspec:cMelspec]
reader.dmLevel=fftmag
writer.dmLevel=melspec
copyInputName = 1
processArrayFields = 1
; htk compatible sample value scaling
htkcompatible = 1
nBands = 26
; use power spectrum instead of magnitude spectrum
usePower = 1
lofreq = 0
hifreq = 8000
specScale = mel
inverse = 0

[mfcc:cMfcc]
reader.dmLevel=melspec
writer.dmLevel=ft0
copyInputName = 1
processArrayFields = 1
firstMfcc = 0
lastMfcc  = 12
cepLifter = 22.0
htkcompatible = 1


[delta:cDeltaRegression]
reader.dmLevel=ft0
writer.dmLevel=ft0de
nameAppend = de
copyInputName = 1
noPostEOIprocessing = 0
deltawin=2
blocksize=1

[accel:cDeltaRegression]
reader.dmLevel=ft0de
writer.dmLevel=ft0dede
nameAppend = de
copyInputName = 1
noPostEOIprocessing = 0
deltawin=2
blocksize=1

  ////////////////////////////////////////////////////////////////////
 ////////////////////  data output configuration  ////////////////////////
//////////////////////////////////////////////////////////////////////

; the HTK sink writes data in HTK parameter format
[htkout:cHtkSink]
 ; data from the following dataMemory levels in concattenated
reader.dmLevel=ft0;ft0de;ft0dede
 ; this again defines a commandline option for the output file (see waveIn)
filename=\cm[output(O){mfcc.htk}:name of MFCC output filename (HTK format)]
append = 0
 ; MFCC_0_D_A  6+256+512+8192 = 8966
parmKind = 8966


//////---------------------- END -------------------------///////
```

Si se lee este fragmento del fichero de configuración, se puede ver que es una programación por bloques, que se alimentan unos a otros, por ejemplo, el bloque cMelspec escribe su salida

en el nivel llamado melspec, y el bloque cMfcc coge como entrada el nivel melspec y escribe en otro nivel llamado ft0.

Como se ha comentado antes también se puede obtener la salida en formato Arff, para ello lo que se tiene que hacer es, en lugar de tener el bloque cHtkSink poner el siguiente bloque:

```
[arffSink:cArffSink]
reader.dmLevel = ft0;ft0de;ft0dede
filename = /home/gtc/cvillalta/opensmile/data_out/mfCc_0sa1.arff
append = 0
relation = mfcc
number = 0
timestamp = 0
frameIndex = 0
frameTime = 0

class[0].name = class
class[0].type = {trama}
target[0].all = \cm[label1{trama}:instance nominal class label]
```

A todas las variables que están declaradas en este fichero de configuración se les puede dar otro valor diferente que se prefiera para calcular las parametrizaciones. Para encontrar una información más detallada de cada uno de los bloques que se pueden implementar así como de sus variables mirar en [19].

Anexo III. Uso de redes neuronales en MATLAB

Matlab posee una toolbox para trabajar con redes neuronales artificiales. Para saber cuáles son y cómo trabajan las funciones que crean, entrenan y testean las redes neuronales vamos a mostrar ejemplos de cómo utilizarlas.

1. Para crear una red neuronal backpropagation con alimentación hacia delante utilizamos la función newff(). Le introducimos como parámetros de entrada el número de capas y el número de neuronas en cada capa; las funciones de activación de cada capa; la función de entrenamiento; y por último le introducimos los valores máximo y mínimo de nuestros datos:

```
net=newff(minmax(datos),layers,functions,funtrain);
```

   donde

```
layers=[2 2 2];
functions = {'tansig' 'tansig' 'logsig'};
%elegimos 'logsig' como función de salida porque queremos
que los valores estén entre 0 y 1.
funtrain='trainlm';
```

2. El segundo paso es entrenar la red neuronal con la función train(), pero para este paso solo utilizamos una pequeña parte de los datos, porque este paso se puede ver como una inicialización de la red.

```
[net tr]=train(net,[datos_0t;datos_1t]',clasest');
```

   como se puede observar introducimos los datos y sus etiquetas (clasest).

3. El siguiente paso es entrenar la red con todos los demás datos, para ello utilizamos adapt(), vamos a introducirle los datos en bloques de 20 en 20.

```
INC = 20;
for i=1:INC:length(datos)
iend = min(length(datos),i+INC-1);
```

```
[net,out_i,error_i,pf_i]=adapt(net,datos(:,i:iend),clase(:,
i:iend));
end
```

4. Por último, para testear la red, utilizamos la función sim(), se ejecuta tantas veces como clases tengamos que clasificar. Y después con los resultados que nos devuelve se calcula la tasa de acierto para los datos de test.

```
results_0 = sim(net,datatest_0');
```

Anexo IV.PCA

El PCA (Principal Component Analysis) o el análisis de componentes principales construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos. En el cual la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (llamado la Primera Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente. Para construir esta transformación lineal debe construirse primero la matriz de covarianza o matriz de coeficientes de correlación. Debido a la simetría de esta matriz existe una base completa de vectores propios de la misma. La transformación que lleva de las antiguas coordenadas a las coordenadas de la nueva base es precisamente la transformación lineal necesaria para reducir la dimensionalidad de los datos.

Una de las ventajas de PCA para reducir la dimensionalidad de un grupo de datos, es que retiene aquellas características del conjunto de datos que contribuyen más a su varianza, manteniendo un orden de bajo nivel de las componentes principales e ignorando las de alto nivel. El objetivo es que esas componentes de bajo orden a veces contienen el "más importante" aspecto de esa información.

Método basado en las covarianzas

Este método tiene como objetivo transformar un conjunto dado de datos **X** de dimensión *n x m* a otro conjunto de datos **Y** de menor dimensión *n x l* con la menor pérdida de información útil posible utilizando para ello la matriz de covarianza.

Se parte de un conjunto *n* de muestras cada una de las cuales tiene *m* variables que las describen y el objetivo es que, cada una de esas muestras, se describa con solo *l* variables, donde $l < m$. Además, el número de componentes principales *l* tiene que ser inferior a la menor de las dimensiones de X,

$$l \leq min\{n, m\} \qquad (IV.1)$$

Los datos para el análisis tienen que estar centrados a media 0 (restándoles la media de cada columna) y/o autoescalados (centrados a media 0 y dividiendo cada columna por su desviación estándar).

$$X = \sum_{a=1}^{l} t_a p_a^T + E \quad (IV.2)$$

Los vectores t_a son conocidos como *scores* y contienen la información de cómo las muestras están relacionadas unas con otras, además tienen la propiedad de ser ortogonales. Los vectores p_a^T se llaman *loadings* e informan de la relación existente entre las variables y tienen la cualidad de ser ortonormales. Al coger menos componentes principales que variables y debido al error de ajuste del modelo con los datos, se produce un error que se acumula en la

matriz E.El PCA se basa en la descomposición en vectores propios de la matriz de covarianza. Se calcula de la siguiente manera:

$$cov(X) = \frac{X^T X}{n-1} \qquad (IV.3)$$

$$cov(X)p_a = \lambda_a p_a \qquad (IV.4)$$

$$\sum_{a=1}^{m} \lambda_a = 1 \qquad (IV.5)$$

donde λ_a es el valor propio asociado al vector propio p_a. Por último,

$$t_a = X p_a \qquad (IV.6)$$

Esta ecuación la podemos entender como que t_a son las proyecciones de X en p_a, donde los valores propios λ_a miden la cantidad de varianza capturada, es decir, la información que representan cada una de las componentes principales. La cantidad de información que captura cada componente principal va disminuyendo según su número, es decir, la componente principal número uno representa más información que la dos y así sucesivamente.

Anexo V. Distribuciones de probabilidad

V.1 Distribución Beta

La distribución Beta es una distribución de probabilidad continua con dos parámetros a y b, cuya función de densidad para valores 0<x<1 es

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} \qquad (V.1)$$

Donde Γ es la función gamma.

El valor esperado y la varianza de una variable aleatoria X con distribución Beta son

$$E[X] = \frac{a}{a+b} \qquad (V.2)$$

$$var[X] = \frac{ab}{(a+b+1)(a+b)^2} \qquad (V.3)$$

Un caso especial de la distribución Beta con a=1 y b=1 es la distribución uniforme en el intervalo [0,1].

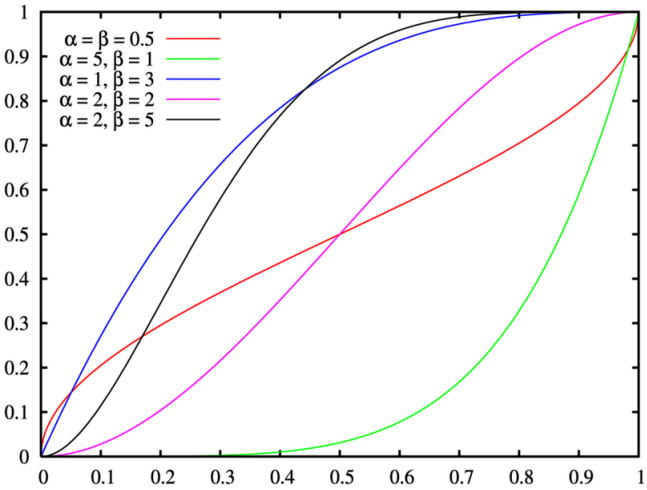

Figura V.1. Función de densidad de probabilidad Beta

V.2 Distribución de Bernoulli

La distribución de Bernoulli o dicotómica es una distribución de probabilidad discreta, que toma valor 1 para la probabilidad de éxito (p) y valor 0 para la probabilidad de fracaso (q=1-p).

Si X es una variable aleatoria que mide "número de éxitos", y se realiza un único experimento con dos posibles resultados (éxito o fracaso), se dice que la variable aleatoria X se distribuye como una Bernoulli de parámetro p.

$$X \sim Be(p) \qquad (V.4)$$

La fórmula será:

$$f(x) = p^x(1-p)^{1-x} \quad con\ x = \{0,1\} \qquad (V.5)$$

Su función de probabilidad viene definida por:

$$f(x;p) = \begin{cases} p & si\ x = 1, \\ q & si\ x = 0, \\ 0\ en\ cualquier\ otro\ caso \end{cases} \qquad (V.6)$$

Un experimento al cual se aplica la distribución de Bernoulli se conoce como Ensayo de Bernoulli o simplemente ensayo, y la serie de esos experimentos como ensayos repetidos.

El valor esperado y la varianza de una variable aleatoria X con distribución de Bernoulli son

$$E[X] = p \qquad (V.7)$$

$$var[X] = p(1-p) = pq \qquad (V.8)$$

V.3 Redes Bayesianas

Formalmente, las redes Bayesianas son gráficos acíclicos dirigidos cuyos nodos representan variables y los arcos que los unen codifican dependencias condicionales entre las variables. Los nodos pueden representar cualquier tipo de variable, ya sea un parámetro medible, una variable latente o una hipótesis.

Si existe un arco que una un nodo A con otro nodo B, A es denominado un padre de B, y B es llamado hijo de A. El conjunto de nodos padre de un nodo Xi se denota como padres(Xi), Un gráfico acíclico dirigido es una red Bayesiana relativa a un conjunto de variables si la distribución conjunta de los valores del nodo puede ser escrita como el producto de las distribuciones locales de cada nodo y sus padres:

$$\mathrm{P}(\mathrm{X_i}, \dots, \mathrm{X_n}) = \prod_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{P}\big(\mathrm{X_i}|\mathrm{padres}(\mathrm{X_i})\big) \qquad (V.9)$$

Si el nodo Xi no tiene padres, su distribución local de probabilidad se toma como incondicional, en otro caso es condicional. Si el valor de un nodo es observable, dicho nodo es un nodo de evidencia.

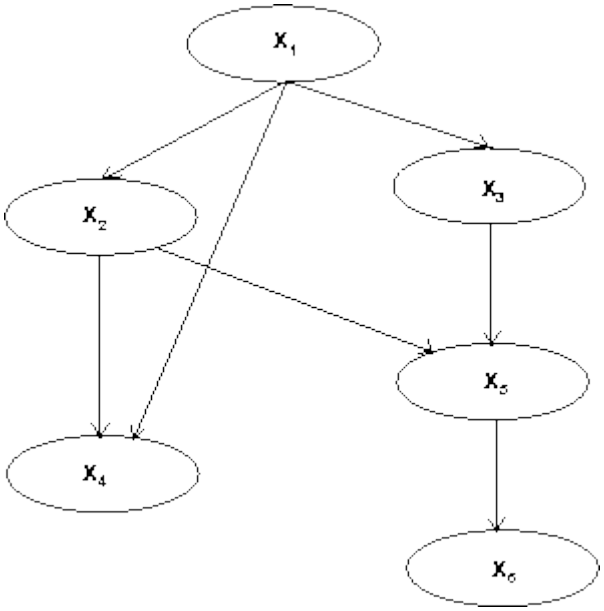

Figura V.2. Ejemplo de una red Bayesiana.

Anexo VI. HTK streams

HTK trabaja principalmente con modelos de densidades de probabilidad continuas, en concreto con mezclas de Gaussianas. En este caso, para el estado j la probabilidad $b_j(o_t)$ de generar la observación o_t viene dada por:

$$b_j(o_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_{js}} c_{jsm} N\left(o_{st}; \mu_{jsm}, \Sigma_{jsm}\right) \right]^{\gamma_s} \qquad (VI.1)$$

donde M_{js} es el número de componentes de la mezcla en el estado j para el stream s, c_{jsm} es el peso de la componente m y $N\left(o_{st}; \mu_{jsm}, \Sigma_{jsm}\right)$ es una Gaussiana multivariante con vector de medias μ_{jsm} y matriz de covarianza Σ_{jsm}, esto es:

$$N(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)' \Sigma^{-1} (o-\mu)} \qquad (VI.2)$$

donde n es la dimensión de o. El exponente γ_s es el peso del stream y su valor por defecto es uno. Pueden utilizarse otros valores distintos de uno para enfatizar un particular stream, pero el HTK estándar no lo puedo realizar.

En este proyecto se van a utilizar no solo el modelo de mezclas de Gaussianas sino también otro tipo de modelos mezclado con el primero, esta manera de trabajar esta implementada en el reconocedor HTK_STUZ, reconocedor implementado por el grupo GTC del I3A. Ahora para el estado j la probabilidad $b_j(o_t)$ de generar la observación o_t viene dada por:

$$b_j(o_t) = \sum_{m=1}^{M_j} c_{jm} \left[N\left(o_{t(1-39)}; \mu_{jm}, \Sigma_{jm}\right) \right]^{\gamma_1} \left[f_2\left(o_{t(40-45)}; \theta_2\right) \right]^{\gamma_2} \quad (VI.3)$$

donde $f_2\left(o_{t(40-45)}; \theta_2\right)$ es un modelo estadístico diferente a la mezcla de Gaussianas. Podemos observar en la ecuación VI.3 que la observación o_t se divide en dos partes: las 39 primeras dimensiones se utilizan en las mezclas de Gaussianas y el resto, cuya dimensión puede variar se utiliza en el otro modelo. Y con los pesos γ_1 y γ_2 podemos enfatizar más un modelo u otro.

Anexo VII. Listado de fonemas

Los fonemas que se han clasificado en el presente proyecto son:

Vocales

/ey/	ABRAHAM	**ey** b r ah hh ae m
/ea/		
/eh/	AHEAD	ah hh **eh** d
/ih/	BEARD	b ih r d
/ao/	ALEATORY	ey l iy ah t **ao** r iy
/ae/	BOOMERANG	b uw m er **ae** ng
/aa/	BORROW	b **aa** r ow
/oh/		
/uw/	BEAUTIFUL	b y **uw** t ah f ah l
/ua/		
/uh/	RAYBOURN	r ey b **uh** r n
/er/	REACTOR	r iy ae k t **er**
/ay/	BEDSIDE	b eh d s **ay** d
/oy/	REJOIN	r iy jh **oy** n
/iy/	BEANS	b **iy** n z
/ia/		
/aw/	INBOUND	ih n b **aw** n d

/ow/	BEAU b **ow**
/ax/	
/ah/	SAXON s ae k s **ah** n

Tabla VII.1. Vocales

Nasales

/m/	SCHEMATIC s k ih **m** ae t ih k
/n/	SCIENCE s ay ah **n** s
/ng/	SCORING s k ao r ih **ng**

Tabla VII.2. Nasales

Africadas

/ch/	SCOTCH s k aa **ch**
/jh/	SERGENT s er **jh** ah n t

Tabla VII.3. Africadas

Fricativas

/s/	SERIAL **s** ih r iy ah l
/sh/	SESSION s eh **sh** ah n
/z/	SESSION'S s eh sh ah n **z**
/zh/	SUBDIVISION s ah b d ih v ih **zh** ah n
/f/	SUBSURFACE s ah b s er **f** ah s
/v/	SUBVERSION s ah b **v** er zh ah n
/th/	SUDDETH s ah d ih **th**
/dh/	TEETH t iy **th**

Tabla VII.4. Fricativas

Líquidas

/l/	TELECARD t eh **l** ah k aa r d
/el/	
/r/	ELABORATE ih l ae b **r** ah t
/ua/	
/ia/	
/w/	IMPETUOUS ih m p eh ch **w** ah s
/y/	IMPUGNED ih m p **y** uw n d
/hh/	INCOHERENCE ih n k ow hh ih r ah n s

Tabla VII.5. Líquidas

Oclusivas

/p/	INDEPENDENCE ih n d ih *p* eh n d ah n s
/b/	INDIVISIBLE ih n d ih v ih s ih **b** ah l
/t/	INDOLENT ih n d ah l ah n **t**
/d/	INDOOR ih n **d** ao r
/k/	INDUCTANCE ih n d ah **k** t ah n s
/g/	INGREDIENT ih n **g** r iy d iy ah n t

Tabla VII.6. Oclusivas

Nota: en los fonemas que no hay un ejemplo, esto es debido a que no hay palabras en el diccionario que se utiliza para reconoce , que tengan estos fonemas.