Universidad Zaragoza
1542

Proyecto Fin de Carrera

Ingeniería de Telecomunicaciones

Sistema de clasificación de sonidos de la vida diaria

Autor

Gonzalo Blasco Soro

Director

José Ramón Beltrán Blázquez

Escuela de Ingeniería y Arquitectura

Marzo de 2015

Agradecimientos

Me gustaría agradecer en primer lugar a José Ramón Beltrán por no sólo guiarme y ayudarme durante el transcurso del proyecto sino también orientarme siempre que lo he necesitado. Ha sido todo un placer desarrollar el proyecto y aprender con él y de él, y compartir numerosas charlas de toda índole.

De la misma manera, agradecer a David Buldain por dedicar gran parte de su tiempo a enseñarme cómo enfocar el uso de redes neuronales de cara al proyecto y por preocuparse constantemente por el desarrollo de éste, ofreciéndome hasta el último momento posibles nuevas mejoras y su experiencia.

Quiero agradecer también a mi familia por todo su apoyo durante estos casi 6 años de carrera y por brindarme la oportunidad de irme un año de Erasmus. A mi abuelo por aprenderse de memoria el nombre de las asignaturas más difíciles y preguntarme por ellas constantemente cada vez que hemos hablado.

Igualmente me gustaría dar las gracias a mis amigos, tanto a los de siempre como a los que he ido conociendo en el camino, que han hecho de estos 6 años un proceso mucho menos monótono e infinitamente más ameno y lleno de experiencias irrepetibles.

Por último agradecer a la música que tantas amistades, bandas y satisfacciones me ha regalado, espero que lo siga haciendo por mucho tiempo.

Resumen

Este proyecto tiene como objetivo principal el reconocimiento y clasificación de diferentes sonidos asociados a la actividad de la vida cotidiana. Este campo de estudio tiene aplicaciones potenciales en el ámbito del reconocimiento de contextos y muy especialmente en las tecnologías de apoyo a personas con algún tipo de dependencia.

Los sonidos de la vida diaria pertenecen a un tipo de sonidos denominado audio no estructurado, por lo que se han investigado las características que mejor definen estos sonidos de cara a su posterior clasificación. Además ha sido necesario crear una base de datos con las 11 clases de sonido empleadas en el proyecto, a saber “pasos”, “ducha”, “retrete”, “grifo”, “calle”, “coche”, “calle con tráfico”, “lavar vajilla”, “cocinar”, “microondas” y “estornudar”.

De todo el conjunto posible de características que se pueden extraer de un sonido se ha decidido obtener 25: las 14 primeras componentes MFCC, 6 parámetros correspondientes a una descomposición en funciones de Gabor con *matching pursuit* y diversas características temporales y frecuenciales.

Una vez completada la base de datos se ha realizado una clasificación basada en redes neuronales utilizando mapas auto-organizados, desarrollándose diversas mejoras basadas en la evaluación del coeficiente Kappa, el coste computacional y la matriz de confusión, que han dado lugar a una disminución del número total de características a 17 y un clasificador final basado en 4 mapas.

A partir de estos 4 mapas se han creado una serie de tablas para hacer corresponder cada neurona con una clase y una serie de sistemas de optimización y evaluación del clasificador basadas en enventanado temporal e histogramas. Por último se han implementado dos simulaciones del funcionamiento del sistema en tiempo real obteniéndose buenos resultados.

Tabla de contenido

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

En este capítulo haremos un breve resumen de la memoria, centrándonos primeramente en los motivos que nos han llevado a elegir este campo de estudio y en cuáles son nuestros objetivos para el desarrollo de este proyecto. Posteriormente haremos una definición del concepto de sonido no estructurado, el cual nos va a acompañar durante todo el desarrollo del proyecto; y una descripción de las herramientas utilizadas. Por último repasaremos la estructura de la memoria.

1.1 Motivación

Para empezar explicaremos el motivo que nos lleva a centrarnos en el reconocimiento de sonidos de la vida diaria. Este tipo de sonidos, entre los cuales podríamos incluir clases específicas como 'andar en la calle', 'montar en coche', 'ir al baño' o 'darse una ducha' entran dentro de la categoría de sonidos ambientales, un conjunto de sonidos que se quieren reconocer con la intención de saber en el contexto en el que un usuario se encuentra en un momento determinado, comúnmente conocido como *context awareness* o conocimiento del contexto. Reconocer el contexto en el que un conjunto de sonidos ambientales se ha producido tiene aplicaciones de gran relevancia en el campo de la navegación, las tecnologías de apoyo a personas con discapacidad y otros servicios sobre plataformas móviles.

La literatura nos ofrece ejemplos de cada uno de estos casos, como se menciona posteriormente en el apartado '2.1.1 *Context awareness*'. En concreto, el caso de las tecnologías de apoyo a las personas con discapacidad es el que más nos preocupa, debido al crecimiento exponencial que están teniendo las *health smart homes* (domótica asistiva y automatización del hogar para ancianos): la domótica se está convirtiendo en la opción viable para los ancianos y discapacitados que prefieren quedarse en sus hogares a tener que trasladarse a un centro de atención médica o una residencia para personas mayores.

Este concepto se ha desarrollado mucho en los últimos años y engloba el diseño de viviendas inteligentes para mejorar las condiciones de vida e independencia de las personas con falta de autonomía mediante la creación de robots de cuidados o la implementación de sensores en ropa o en los muebles de la casa [1]. Otro importante motivo por el cual es preferible centrarse en este caso es el hecho del inminente envejecimiento de la población debido a la disminución tanto de la tasa de mortalidad como la de natalidad [2].

Además de las potenciales aplicaciones mencionadas anteriormente, hay un motivo muy claro por el que nos hemos adentrado en este campo. El conjunto de sonidos ambientales que este campo de investigación ocupa entra dentro de la categoría de los llamados sonidos no estructurados, esto es, los sonidos que no siguen un patrón concreto de repetición o caracterización como pueden ser la música o el habla. Lo que precisamente le da una perspectiva diferente a este proyecto respecto a otros reconocedores es el hecho de que la investigación en este campo no está tan avanzada como en el caso de reconocedores de voz o de música, pero sin embargo las tecnologías y algoritmos para extraer características espectrales y temporales de tramas de audio están ya desarrollados y estudiados debido precisamente a éstas.

Un estudio exhaustivo de las diferencias entre sonidos no estructurados y estructurados nos puede llevar a conseguir un sistema de detección y clasificación de sonidos no estructurados con relativa facilidad en cuanto a los medios tecnológicos se refiere. Sin embargo no es tan fácil si tenemos en cuenta la gran aleatoriedad y variedad de estos sonidos, que suelen estar solapados o superpuestos entre ellos. Además son fuentes con una gran cantidad de ruido. Este será un factor condicionante en la calidad de nuestro reconocedor, y por tanto limitará el número de clases que podemos distinguir con él.

El uso de redes neuronales para clasificar fuentes de audio no estructurado es también un aspecto innovador introducido en este proyecto, ya que por lo general se hace uso de otros algoritmos para la clasificación, los cuales comentaremos en el capítulo dedicado al estado del arte.

1.2 Objetivos

El objetivo principal de este proyecto es implementar un sistema clasificador de diversas clases de audio no estructurado con sonidos de la vida diaria y simular su funcionamiento.

Para ello, se hará un estudio sobre cuáles son las características espectrales y temporales que mejor definen al sonido no estructurado y, una vez creada una base de datos con distintas fuentes de audio, se hará la extracción de características para crear un sistema clasificador basado en un mapa auto-organizado. Tras evaluar el comportamiento del mapa se implementará una simulación de un evento en tiempo real, la cual nos dará un resultado sobre la calidad de nuestro reconocedor. Para conseguir refinar esa calidad se harán posteriores estudios donde se irán suprimiendo características y buscando una longitud de ventana de audio hasta encontrar una combinación que nos dé un resultado óptimo. Las características eliminadas serán por tanto las más ruidosas y las que se queden serán las más importantes a la hora de elegir entre el conjunto de clases de sonido seleccionado.

1.3 ¿Qué es el sonido no estructurado?

El sonido no estructurado (*unstructured sound* en inglés) es un término derivado de la comparación de las características que definen a los sonidos ambientales, como pueden ser el sonido de la calle, los coches pasando, o el sonido de unos pasos; con respecto a las características que definen otros tipos de sonido más intencionado como son el habla o la música.

La diferencia básica entre los sonidos no estructurados y los estructurados radica en su naturaleza, dado que los no estructurados contienen una variedad inmensa de sonidos con picos espectrales muy estrechos o *chirping*, característica difícil de localizar mediante las técnicas habituales para sonidos estructurados, además de la aleatoriedad y gran variabilidad que es inherente [3]. Puede decirse que las características de los eventos de sonido no estructurado difieren de las del habla principalmente en la riqueza de contenido en frecuencia, la duración y el perfil.

Además, para el caso de el sonido no estructurado no existe ningún tipo de subdiccionario de señales de audio equivalente al de la descomposición en fonemas de la señal

de voz. Por ejemplo, a la hora de la descomposición de una señal de voz, podríamos descomponer la palabra "proyecto" en los fonemas /p/, /r/, /o/, /j/, /e/, /k/, /t/ y /o/, o en grupos de fonemas o sílabas, pero ese no es el caso del reconocimiento de sonidos no estructurados, ya que contaríamos con un diccionario excesivamente variado de todos los posibles ruidos que conforman cada una de las clases de audio. Por último el ruido, la distorsión y la superposición de sonidos ocurre muy a menudo en los contextos donde los sonidos ambientales tienen lugar, haciendo el reconocimiento de estos una tarea más compleja.

1.4 Herramientas utilizadas

1.4.1 Audacity

Audacity es un editor de grabación y edición de sonido libre, de código abierto y multiplataforma. Su uso varía desde la grabación de audio en vivo o estudio hasta la edición y conversión entre distintos formatos de sonido. Es frecuentemente utilizado para la mezcla de distintas pistas de audio debido a sus múltiples *plugins* de ecualización y compresión, además de los efectos de sonido, desarrollados por programadores de todo el mundo [4].

Audacity no cuenta con las mismas funciones, algoritmos y potencia que los softwares equivalentes de uso comercial, como pueden ser Pro Tools o Cubase, pero es una herramienta más que útil para iniciarse en la edición y mezcla de sonido y por supuesto para la finalidad con la que se ha usado en este proyecto: el recorte y etiquetado de pistas de audio y su conversión a una frecuencia de muestreo común de 22.05 kHz.

1.4.2 Matlab

Matlab es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación usado por millones de ingenieros y científicos en todo el mundo [5]. Mediante *Matlab* es posible analizar datos, desarrollar algoritmos y crear modelos o aplicaciones. El lenguaje, las herramientas y las funciones matemáticas incorporadas permiten explorar diversos enfoques y llegar a una solución antes que con hojas de cálculo o lenguajes de programación habituales. *Matlab* es utilizado en una gran variedad de aplicaciones tales como procesamiento de señal y comunicaciones, sistemas de control, pruebas y medidas, finanzas computacionales, biología computacional, procesamiento de imagen y video, o el caso que nos concierne, procesamiento de audio.

En este proyecto *Matlab* ha jugado un papel muy importante al ser la herramienta principal sobre la cual se desarrolla todo el preprocesado, análisis y la posterior clasificación del audio, así como la implementación de la simulación de un reconocedor de secuencias de eventos y la eliminación de variables irrelevantes para mejorar la eficacia del clasificador. Además, la mayor parte de las gráficas, figuras e histogramas obtenidos para esta memoria también han sido generadas con *Matlab.*

Aparte de usar las funcionalidades más básicas de *Matlab*¸ también se ha hecho uso de una librería llamada *somtoolbox* [6] de la cual se hablará más en detalle durante el transcurso de la memoria. Mediante el uso de esta librería hemos pretendido hacer una clasificación de los distintos tipos de fuentes con el uso de una red neuronal y un mapa auto-organizado.

1.5 Estructura de la memoria

La memoria sigue un discurso que se asemeja bastante a la cronología de eventos durante el desarrollo del proyecto desde su comienzo en Septiembre de 2014. Así pues nos centraremos primero en una revisión del estado del arte, después en el desarrollo del proyecto en sí y después en el refinamiento de los resultados mediante el empleo de una simulación de una secuencia de eventos, la eliminación de características redundantes y el agrupamiento de sonidos en subclases. Por último presentaremos nuestras conclusiones al respecto y las líneas futuras para seguir con el desarrollo en otros proyectos.

Capítulo 2: Estado del arte

En este capítulo vamos a explicar las implementaciones que se han hecho hasta la fecha basadas en algunos conceptos relacionados con el audio no estructurado y las características de nuestra base de datos. Posteriormente se citan los usos típicos en lo que concierne al reconocimiento de audio no estructurado y una breve explicación de para qué se usan por norma general las características temporales y espectrales que hemos usado para hacer el clasificador. Para finalizar el capítulo comentaremos los clasificadores típicos y las principales diferencias entre ellos.

2.1 Sonidos no estructurados

Como comentábamos anteriormente en el apartado de motivación, dentro del campo de reconocimiento de sonidos la parte correspondiente a sonidos ambientales y no estructurados ha recibido menor atención que la orientada al reconocimiento del habla y de patrones musicales. Esto es debido mayoritariamente a la variabilidad de las señales, su naturaleza, las superposiciones y la dificultad a la hora de encontrar subdiccionarios que las caractericen por completo.

Aún así, hay una buena colección de artículos científicos dedicados a este campo y en ellos nos hemos basado a la hora de elegir el conjunto de características y el método de trabajo.

Las aplicaciones relacionadas con el audio no estructurado han sido principalmente desarrolladas en el campo de *context awareness*:

2.1.1 *Context awareness*

Dentro del campo de ciencias de la computación, el concepto de *context awareness* se refiere a la idea de que los ordenadores (o microprocesadores, microcontroladores, etc.) puedan sentir y reaccionar basados en el ambiente y contexto que los rodea. Así pues los dispositivos deben de tener una información correspondiente al estado normal de las cosas y reaccionar en consecuencia cuando aparezca un evento anormal basándose en un conjunto de reglas previamente diseñadas. El término fue acuñado por Schilit en 1994 [7] y ha ido evolucionando con el tiempo, hasta el punto en el que los propios dispositivos son capaces de hacer suposiciones sobre el estado actual del usuario y reaccionar en consecuencia de una manera previamente programada.

En algunos artículos como el [8] se habla de el reconocimiento de audio como ayuda para la navegación y saber en qué tipo de contexto se encuentra un robot en un instante determinado; en el caso de [9] también se hace uso de este aspecto de la tecnología, tratando de evitar la interacción humana para saber en qué tipo de contexto está en cada momento un teléfono móvil y actuar en consecuencia. Por último en el caso de [10] el concepto se extiende hasta para ser usado en las llamadas *health smart homes*, o lo que es lo mismo “casas inteligentes” con ciertos sensores monitorizando la actividad de personas con dependencia.

Estas y otras aplicaciones dentro del campo de *context awareness* han sido combinadas con la respuesta enviada desde otro tipo de sensores como los infrarrojos, cámaras de video o etiquetas RFID, dando lugar a una mayor efectividad a la hora de hacer el reconocimiento de actividades. También cabe mencionar que se han creado ciertos indicadores del grado de dependencia que necesita una persona, como puede ser el Índice creado por Katz y Akpom en [11], el cual refleja el grado de independencia de una persona basándose en la habilidad de ésta para levar a cabo ciertas actividades clave de la vida diaria.

2.1.2 Bases de datos

Por otro lado y debido a la expansión tanto de internet como de las licencias públicas de copyright tipo *creative commons* [12] es posible contar con una base de datos lo suficientemente grande como para poder hacer una selección previa de nuestro conjunto de fuentes de audio. En concreto hemos trabajado con parte de la base de datos de la página web *freesound.org* [13], desarrollada por el departamento de *Music Technology* de la Universidad Pompeu y Fabra de Barcelona, donde es posible encontrar todo tipo de grabaciones desde pistas de música y de habla hasta sonidos ambientales o efectos sonoros, todo ello clasificado por diversos parámetros como la frecuencia de muestreo, el *bitrate* o el formato de archivo. El hecho de usar licencias *creative commons* hace posible la utilización de estas fuentes de sonido sin verse involucrado en problemas legales al hacerlo, y por supuesto nos ha ahorrado la larga tarea de hacer nuestras propias grabaciones.

2.2 Selección de características

Para empezar, el hecho de que haya una diferenciación entre el sonido estructurado y el no estructurado no es infundado ni mucho menos: las propiedades espectrales y temporales que caracterizan a uno y otro tipo de sonidos son diferentes, y de ahí la separación. No se trata sólo de lo que el oído humano percibe (un sonido más tipo ambiental o un sonido más 'ordenado', música o voz hablada) sino del tipo de características que tiene cada una de estas clases de sonido.

La extracción de características de un sonido se hace posteriormente a la división de éste en *frames* o ventanas, normalmente de longitud una potencia de 2. A partir de éste *frame* es desde donde se empieza a trabajar para caracterizar el sonido. Para evitar efectos de borde se introduce un solapamiento u *overlapping* entre ventana y ventana. Esto permite además el poder trabajar en tiempo real, ya que es imposible trabajar con la totalidad de una señal de audio cuando aún está siendo grabada por un micrófono: desde el momento en que se detecta actividad de audio se hace procesado con fragmentos o ventanas de una longitud corta para dar una respuesta rápida.

De la división entre sonidos estructurados y no estructurados podemos concluir que hay características comunes a ambos tipos de sonido y otras más específicas para cada tipo. Hay numerosos artículos científicos que justifican matemáticamente qué tipo de características son las más adecuadas para hacer cada tipo de reconocimiento. A continuación haremos un breve repaso sobre el estado del arte de las que creemos más relevantes a la hora de hacer la clasificación que nos ocupa en este proyecto.

2.2.1 Mel Frequency Cepstral Coefficients

Los parámetros o características que por lo general mejor suelen definir un tipo de señal de audio genérico son los Mel Frequency Cepstral Coefficients (Coeficientes Cepstrales en las Frecuencias de Mel, de ahora en adelante MFCC). Éstos son utilizados en la gran mayoría de reconocedores de sonido hoy en día y fueron utilizados por primera vez en 1980 por Davis y Mermelstein [14]. Desde entonces se han consolidado como una de las mejores características a la hora de hacer reconocimiento de audio, ya sea por separado o combinados con otras características. El uso de las primeras y segundas derivadas de estos coeficientes como característica también ha sido ampliamente utilizado para definir su cambio a lo largo del tiempo o de un *frame* al siguiente, como hacen Zhu y Alwan en [15] para emplearlo en el reconocimiento de habla. También han sido empleados como característica para hacer un reconocimiento del instrumento musical que está tocando en publicaciones como [16], donde además se hace un estudio de cuál es la cantidad óptima de coeficientes que se deben elegir a la hora de hacer una clasificación.

2.2.2 Características basadas en *matching pursuit*

Otro tipo de características muy importantes a la hora de clasificar sonido, y en específico, sonido no estructurado, son las basadas en *matching pursuit*. *Matching pursuit* es una aproximación que busca las proyecciones que mejor acercan un conjunto de datos con un diccionario predeterminado o base, obteniendo un error que decrece exponencialmente según va aumentando el número de iteraciones. Este algoritmo es usado para codificar y comprimir audio, video e imágenes, entre otros usos. La idea es pues representar nuestro *frame* actual como la suma ponderada de unas funciones o átomos, obviamente escogiendo los más significativos de manera que la energía de la señal de error sea la mínima posible. Si escogiéramos un diccionario de señales senoidales tendríamos por tanto una aproximación a la transformada de Fourier, pero diversos artículos como [17] o [3] sugieren que lo mejor es escoger un diccionario de funciones que varíen en frecuencia y en tiempo, como es el caso de las funciones de Gabor. Esto es debido a que la variación en frecuencia caracterizará mejor una zona espectral y la variación temporal otra, complementándose a la hora de crear un diccionario de funciones que se comporte bien en todo el espectro de interés.

Las funciones de Gabor son unas funciones de envolvente gaussiana moduladas senoidalmente, escaladas y desplazadas en tiempo. Según [3] la media y la varianza de la frecuencia, desplazamiento y escala de los primeros átomos que comprenden la descomposición según *matching pursuit* de un *frame* nos darán las características que mejor lo representan. De hecho es en este estudio donde además se recomienda no escoger más de 5 átomos para representar un fragmento de audio.

2.2.3 Otras características

Además de las dos características mencionadas anteriormente, hay muchas otras que han dado lugar a varias aplicaciones de gran relevancia dentro del ámbito de la clasificación de señales de audio . Uno de estos casos es la localización de constelaciones de picos espectrales dentro de un espectrograma. La famosa aplicación *Shazam,* que consiste en un reconocedor de música a través de la grabadora del teléfono móvil, se basa principalmente en esto.

Mediante esta técnica, donde cada uno de estos picos hace referencia a una posición en tiempo y frecuencia, se ha conseguido llegar a hacer reconocimiento de canciones tomando una grabación de tan sólo 10 segundos, con ruido ambiental incluido, de entre una base de datos de 1.8 millones de pistas [18]. En el caso de este proyecto no será utilizada dado que no es el caso de audio estructurado y ese tiempo entre picos que tan bien define una señal de audio musical no lo hará en el caso de audio ambiental.

Por otro lado hay otras características, muchas de ellas más fáciles de implementar, que combinadas con las dos mencionadas anteriormente nos pueden ayudar a mejorar nuestro sistema clasificador. Entre ellas están los LPC o Coeficientes de Predicción Lineales, el ratio de energía por banda, el factor de roll-off en frecuencia, el centroide, la asimetría y el ancho de banda espectrales, la energía total o el ratio de cruces por cero. Algunas de ellas han sido utilizadas durante el desarrollo del proyecto para ser posteriormente combinadas con los coeficientes MFCC o las características provenientes de *matching pursuit*.

2.3 Clasificación de clases de sonidos

A la hora de hacer la diferenciación entre unos sonidos u otros es muy importante tener en cuenta cuál va a ser la técnica o el algoritmo encargado de interpretar las características correspondientes a cada clase y hacer una separación entre ellas ya que esto puede influir enormemente en el resultado.

Aunque nosotros hemos querido innovar y arriesgar un poco con la selección de las redes neuronales como clasificadoras en nuestro sistema de reconocimiento, hay otras técnicas muy avanzadas con altos índices de reconocimiento. El mayor inconveniente que éstas tienen en contra del sistema descrito en este proyecto es la baja eficiencia computacional del clasificador, la cual se ve muy aumentada con la inclusión de redes neuronales. Describiremos algunas de estas técnicas y sus diferencias a continuación en los subapartados. En muchos casos no se usa sólo una de ellas en concreto, sino un híbrido entre 2 o más técnicas para potenciar la efectividad de la clasificación [19].

Además cabe mencionar que hay dos grandes tipos de clasificadores debido a su naturaleza, a saber, clasificadores de aprendizaje supervisado y clasificadores de aprendizaje no supervisado. En el caso específico del reconocimiento y clasificación de sonido a los clasificadores con aprendizaje supervisado se les da dos tipos de entrada: los datos que caracterizan al sonido que queremos clasificar y el resultado esperado. Así pues el objetivo de un clasificador supervisado es crear una función que sea capaz de responder de la misma manera a entradas más o menos parecidas a las de los ejemplos, para cada una de las clases. Sin embargo, en el caso del aprendizaje no supervisado el modelo se ajusta a las observaciones sin saber a priori qué observaciones pertenecen a una clase o a otra. Por tanto en este caso se trata a todas las observaciones o datos de entrenamiento por igual, y en lugar de tratar de modelar una función que se asemeje a un conjunto de resultados, lo que intentamos es modelar la función de densidad de estos datos. Obviamente necesitamos saber a qué clase de sonido o resultado pertenece cada una de nuestras observaciones para valorar la calidad de nuestro reconocedor, pero no para que éste clasifique *per se*.

2.3.1 Redes neuronales

Empezaremos haciendo una breve descripción del estado del arte en el caso de las redes neuronales ya que es el caso que nos ocupa en este proyecto. El modelo de clasificación basado en mapas auto-organizados que hemos utilizado es un clasificador con aprendizaje no supervisado.

Los *self-organized maps* (comúnmente conocidos como SOM) o mapas auto-organizados fueron desarrollados por *Teuvo Kohonen* en *la Helsinki University of Technology* durante los años ochenta [20] y sirven no sólo como clasificadores sino además como herramienta para ver representaciones multidimensionales de datos en solamente dos dimensiones. Es una técnica considerada como la extensión no-lineal de PCA (*principal component analisis* o análisis de componentes principales) y presenta la gran ventaja de que una vez entrenado el mapa correspondiente a una serie de clases de sonidos lo único que hay que implementar es una *look-up table*: una tabla que nos diga la posición, entendiendo por posición el valor de las componentes que la sitúan en el mapa, de cada una de las neuronas. Cada neurona tiene una clase de sonido asociada y la búsqueda de la neurona que mejor modela una señal de entrada se reduce a simplemente buscar la menor distancia euclídea de entre todas las posibles. Esto computacionalmente es muy eficiente en comparación con otras técnicas de clasificación como las que se mencionan a continuación y ha sido un factor motivador a la hora de elegir la técnica más adecuada para este proyecto.

Las redes neuronales se han utilizado menos que otras técnicas para hacer clasificación de varias clases de sonido, aunque hay referencias a ellas en diversas publicaciones. Es por ejemplo el caso de Hinton [21], el cual hace una combinación de redes neuronales y modelos ocultos de Markov para hacer reconocimiento de habla, esto es reconocer distintos fonemas y su evolución temporal, y posteriormente buscar en un diccionario de palabras aquella que más se adecúe al contexto. En [22] también se hace uso de las redes neuronales, esta vez no sólo con señales de audio sino también con acelerómetros y otros sensores, consiguiendo monitorizar el comportamiento de un individuo en casa diferenciando entre 14 clases distintas con un 95 por ciento de exactitud. Este ejemplo nos ilustra cómo un clasificador puede ser mejorado si además de audio se le añade la entrada de otro tipo de sensores.

Por último cabe comentar que hay otros tipos de redes neuronales que no están basadas en mapas auto-organizados, como pueden ser el mapa auto-organizado incremental o las redes de retropropagación de perceptrones multicapa . Sin embargo diversos estudios como [23] nos muestran cómo SOM es una técnica muy competente para el caso específico de reconocimiento de sonidos, en este caso en concreto para el caso de latidos de corazón.

2.3.2 *Gaussian Mixture Models*

Las *Gaussian Mixture Models* pueden ser empleadas igualmente como clasificadoras y caracterizadoras de distintas subclases de un conjunto de datos de audio, por tanto entraría dentro de los clasificadores de aprendizaje no supervisado. Aunque según ciertas publicaciones y dependiendo del tipo de sonido a clasificar no son el algoritmo a implementar más eficiente, las GMM son la técnica más común con diferencia a la hora de clasificar audio y en concreto a la hora de reconocer voz hablada.

El principal inconveniente de esta técnica es la cantidad de datos de entrenamiento necesaria para que empiece a funcionar de la manera esperada, por lo que es muchas veces rebatida como sistema clasificador [24].

Ha sido utilizada en combinación con Vector Quantization a la hora de hacer reconocimiento de sonido en diversas publicaciones como el reconocimiento automático de especies de aves [31] o en el caso que nos ocupa, reconocimiento de sonidos de la vida diaria, pero esta vez combinado con otras técnicas de clasificación [25].

2.3.3 *K Nearest Neighbours*

Este clasificador es un caso de aprendizaje no supervisado ya que aunque los datos de entrada sean etiquetados con la pertenencia a una clase, esta clase se asigna posteriormente a la clasificación una vez ya se han formado los agrupamientos o *clusters* de datos, como es el caso de SOM.

Es uno de los algoritmos más simples ya que durante la fase de entrenamiento la técnica consiste sólo en almacenar los datos presentados y por tanto la mayor parte de la carga computacional aparece cuando el vector de características es presentado al clasificador, lo cual puede ser un agravante en implementaciones de tiempo real. Aunque se basa en calcular distancias entre el vector presentado al clasificador y los vectores del entrenamiento, lo cual supone poca carga computacional, en general es un tipo de clasificador poco eficiente en comparación con otros en lo que al audio no estructurado se refiere [3].

2.3.4 Modelos Ocultos de Markov

Los modelos ocultos de Markov (*Hidden Markov Models* o HMM) son comúnmente usados para modelar una secuencia de observaciones de una longitud no determinada previamente. Esto sirve de gran ayuda a la hora de modelar la variabilidad de las señales a lo largo del tiempo y es por ello por lo que son ampliamente usados principalmente en el campo del reconocimiento del habla. No es un clasificador como tal, pero sí que ayuda a la hora de hacer reconocimiento de las transiciones de una clase a otra o de la estaticidad en una clase a lo largo del tiempo, por lo que pueden mejorar considerablemente la calidad final de un clasificador.

Un HMM puede entenderse como una generalización de un *mixture model* donde las transiciones de un estado a otro vienen relacionadas por un proceso de Markov en lugar de ser independientes, como es el caso de todas las clases originadas de los distintos clasificadores en esta sección.

2.3.5 *Support Vector Machines*

El ultimo clasificador del que vamos a hacer mención en el apartado del estado del arte es el basado en las *Support Vector Machines*. Se trata de un modelo de aprendizaje supervisado que analiza los datos y reconoce patrones y comúnmente usado para clasificar y hacer modelos de regresión. Su característica principal es que se trata de un clasificador binario, teniendo que hacer una subdivisión del clasificador multiclase en varios clasificadores binarios en forma de diagrama en árbol. Fue propuesto por primera vez por Vapnik en 1995 [26].

Este clasificador ha sido comparado en diversos estudios con otros clasificadores mencionados anteriormente en este apartado, llegando a la conclusión de que se pueden llegar a resultados similares a los obtenidos con GMM ampliando la eficiencia computacional en alrededor de 40 veces cuando se trata de sonido ambiental o no estructurado [27] y [28].

2.4 Eliminación de variables

Ya hemos comentado en el apartado correspondiente a las características que el hecho de tener características irrelevantes de más puede aportar cierto ruido a la hora de hacer la clasificación, empeorando la calidad del clasificador. Hay diversos algoritmos para saber cuál es la combinación de variables que nos dará un factor de reconocimiento máximo.

Primero vamos a hablar de cuál es el índice usado para medir la calidad de clasificación de un conjunto de variables. Se trata del índice Kappa, el cuál va entre 0 y 1, siendo 1 el mejor índice posible para un clasificador. Fue desarrollado por Cohen en 1960 [29]. Un clasificador se empieza a considerar más o menos preciso si su coeficiente kappa supera 0.7, aunque por supuesto hay matices. Su diferencia con respecto a un simple "porcentaje de aciertos en la clasificación" es que tiene en cuenta el efecto del azar en la proporción de la concordancia observada, resultando un coeficiente más robusto y conservador. Éste es pues el índice en el cual nos basaremos a la hora de comparar distintas combinaciones de clases.

Un método muy sencillo y que hemos implementado para saber cómo de influyente o irrelevante es una variable es el ir quitando las variables una a una e ir observando si el clasificador (o el índice) mejora o empeora. Esta es una buena aproximación y puede hacer subir el índice kappa unas cuantas décimas. Sin embargo no tiene en cuenta las no linealidades que pueden derivarse de que por ejemplo dos variables clasifiquen bien juntas pero no por separado. Como para un set largo de *n* variables el conjunto de combinaciones de variables a evaluar es 2^n hemos usado también la herramienta de algoritmos genéticos para comparar.

2.4.1 Algoritmo genético

El algoritmo genético fue introducido por J. Holland en los años setenta y ha supuesto una de las líneas más prometedoras de la inteligencia artificial [30]. Se trata de un sistema de búsqueda heurístico que imita de alguna manera la selección natural, tratando de buscar la mejor solución dentro de una población de potenciales "genomas", reproduciendo los patrones que mejor se comportan y haciendo pequeñas mutaciones para intentar cubrir de alguna manera todo el espacio de búsqueda.

En el caso específico de nuestro proyecto se ha utilizado para intentar suplir a las 2^n combinaciones de variables posibles a la hora de saber cuál es el mejor clasificador posible. Uno de los inconvenientes del algoritmo genético es que puede dar lugar a soluciones no demostrables pero que funcionen mejor que otras alcanzadas por métodos clásicos.

Otras aplicaciones del algoritmo genético son el diseño automatizado de piezas, componentes y materiales industriales, el diseño de tipologías de circuitos o la ingeniería de software, por nombrar unas pocas más cercanas al sector de las telecomunicaciones.

Capítulo 3: Extracción de características y el mapa auto-organizado

En este capítulo haremos una descripción de las diferentes fases del desarrollo del proyecto, en lo que a la construcción del clasificador se refiere. Primero se comentará brevemente las clases de sonido elegidas para ser diferenciadas por el clasificador. Después se hará hincapié en cada una de las fases del preprocesado, justificando cada una de ellas. Posteriormente una descripción completa de las características elegidas para la descripción de cada clase de sonido y por último una explicación sobre el funcionamiento del clasificador y los primeros resultados obtenidos con éste.

3.1 Selección de clases de sonido

Primeramente vamos a comentar qué clases de sonidos hemos elegido para el proyecto. Todos ellos han sido descargados de la base de datos de *The Freesound Project* [13], una base bastante completa y con diversos filtros para poder elegir el tipo de señal deseada. Nos hemos centrado en trabajar con ficheros .wav dado que éstos no están comprimidos, sin centrarnos en la cantidad de bits usados por muestra ya que la resolución en amplitud del audio no nos importa demasiado: nuestras grabaciones contienen bastante ruido y varianza y, eventualmente, en futuros proyectos, el dispositivo grabador será el micrófono de un teléfono móvil o un dispositivo portable similar. Además, cada una de las ventanas va a ser normalizada en amplitud para que el rango dinámico de la señal esté aprovechado al máximo.

El conjunto de clases de sonidos escogidas para la realización del proyecto consta de 11 tipos, a saber “pasos”, “ducha”, “retrete”, “grifo”, “calle”, “interior de un coche”, “calle con tráfico”, “lavar la vajilla”, “cocinar”, “microondas” y “estornudar”. Todos ellos han sido transformados con *Audacity* a una frecuencia de muestreo de 20050 Hz y convenientemente recortados de manera que no haya intervalos de silencio entre sonido y sonido. En total contamos con una base de datos de 212 ficheros con una duración total de 19 minutos y 35 segundos, teniendo entre 9 y 42 señales de distinta longitud de cada tipo de sonido. Hemos intentado que haya más ficheros de los sonidos con menor duración para que la longitud total de cada clase de sonido esté equilibrada con las demás. Estos valores se pueden consultar con más detalle en la Tabla 1.

Además, para tratar de preservar la intimidad del potencial usuario de este sistema, la idea final es implementar un sistema que sólo envíe las características a un servidor y éste sea el que haga la clasificación de sonido, o incluso la posibilidad de tener todo empotrado en el mismo sistema portable. Lo que no hemos contemplado en ningún caso es la posibilidad de mandar la señal de audio a un servidor, precisamente por mantener dicha intimidad.

Clase de sonido	Número de clips	Duración total (mm:ss)
Pasos	42	00:15
Ducha	38	02:29
Tirar de la cadena	10	00:40
Grifo	15	00:59
Calle sin tráfico	22	01:27
Interior de un coche	11	00:44
Calle con tráfico	18	05:06
Lavar la vajilla	18	01:49
Cocinar	15	04:52
Microondas	14	00:51
Estornudar	9	00:19
TOTAL	212	19:35

Tabla 1 - Clases de sonido, clips y duración por clase

3.2 Frecuencia de muestreo, características del enventanado y pre-procesado

A la hora de trabajar con sistemas de clasificación de fuentes de audio en tiempo real hay un compromiso claro entre la velocidad a la que queremos que nuestro sistema clasifique y la cantidad de información obtenida para hacer esa clasificación. Para obtener una información relevante sin por ello obtener un gran *delay* en nuestro sistema podemos dividir la señal de audio de entrada en diferentes *frames* o ventanas de una longitud determinada de muestras de manera que la extracción de características se haga sobre cada una de estas ventanas. Además para según qué características es mejor trabajar con trozos de audio de una longitud corta que con la señal completa. Así pues nosotros haremos lo propio con nuestro sistema de entrenamiento y cada una de las señales que hemos utilizado para entrenar será subdividida según una longitud de ventana y un *overlapping* determinados.

A la hora de saber cuál es la longitud de ventana óptima para realizar el proyecto y entrenar los mapas, hemos tenido que hacer un breve estudio. Esto será explicado con más detalle posteriormente pero en concreto hemos elegido finalmente una longitud de ventana de 2048 muestras y teniendo en cuenta que estamos trabajando con señales muestreadas a 22050 Hz, esto equivale a unos 92.8 ms de audio. El solape entre ventanas es del 50%.

Existe la posibilidad de modificar cada una de las ventanas y a la vez que se hace el recorte de la señal original de audio multiplicarlas o convolucionarlas con un filtro que adapte en cierto modo la ventana a la característica que le vamos a aplicar. Ejemplos de estos filtros son las ventanas de *Hamming*, *Hanning*, *Blackman* o la más sencilla de todas ellas, la ventana rectangular, que deja básicamente la ventana inalterada. Cada una de ellas, excepto esta última, afecta a la amplitud y por tanto a las características espectrales y temporales de la

señal que está siendo tratada en cierto momento. En nuestro caso, hemos usado ventana rectangular en todos los casos excepto en el de la extracción de los MFCCs, donde hemos decidido usar una ventana de *Hamming*.

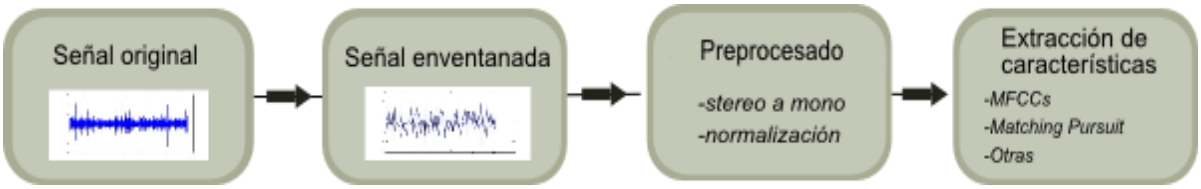

Figura 1 - Diagrama de bloques explicativo del proceso de enventanado y preprocesado

Además ha surgido el problema de tener las distintas señales de entrada con frecuencias de muestreo distintas. Si pretendemos que una variable correspondiente a una característica temporal o espectral no esté sesgada por la frecuencia a la que esta señal ha sido muestreada debemos mantener la misma frecuencia de muestreo para todas las señales de entrenamiento y entrada. Esto se puede solucionar fácilmente con *Audacity*, importando la señal de audio original y cambiando la frecuencia de proyecto a la deseada. El propio programa se encarga de interpolar o diezmar entre muestras, sin por ello afectar a la longitud total en tiempo del fragmento de audio, sino meramente a su calidad y resolución. La frecuencia de muestreo elegida para el proyecto es de 22050 Hz, basándonos en el entorno de trabajo creado en [3]. También hemos decidido normalizar cada uno de los *frames* de manera que cada uno de ellos tenga media cero y desviación estándar igual a 1 entre las 2048 muestras que lo componen y previamente al procesado de las características.

3.3 Selección de características

Tras la lectura de los diversas publicaciones mencionadas en la sección “Capítulo 2: Estado del arte”, nos hemos decidido por centrarnos en una combinación de características basadas en *matching pursuit* con coeficientes MFCC. También hemos incluido alguna característica de implementación más sencilla, de las que se suelen implementar para clasificación de audio genérico.

Un aspecto muy importante y que se subraya en todos los artículos científicos que hablan del reconocimiento y clasificación de sonido es que el conjunto de características que definen una ventana de audio debe estar cuidadosamente elegido, ya que el uso de parámetros potencialmente irrelevantes puede dar lugar a un impacto negativo en la calidad del clasificador. El uso de una o varias características irrelevantes da lugar a un mayor espaciado entre los datos de una misma clase, convirtiéndose estas características en ruido. Por esto mismo se implementará al final del proyecto un algoritmo para eliminar aquellas características que introduzcan ruido o que sean irrelevantes para el sistema clasificador.

Es por tanto muy importante o elegir muy bien y cuidadosamente las características desde el principio o implementar un sistema que nos ayude a eliminar características ruidosas. Hemos optado por esta segunda opción ya que, dependiendo de la naturaleza de los sonidos que queremos saber diferenciar, las características más relevantes pueden ser unas u otras. Dentro del set de características que hemos escogido puede ser que haya alguna que sea especialmente relevante para los sonidos que queremos diferenciar o que sea totalmente

ruidosa. Como a pesar de partir de un set de características ya sesgadas no sabemos hasta qué punto una característica puede influir o no positivamente, haremos un primer estudio con todas ellas.

3.3.1 Características basadas en *matching pursuit*: Funciones de Gabor

Según la literatura, a la hora de reconocer sonidos no estructurados el conjunto de características que mejor van a saber diferenciar una clase de otra está estrechamente relacionado con la descomposición de la señal de audio en la suma ponderada de un conjunto de funciones pertenecientes a un diccionario.

En concreto estamos hablando de un diccionario de funciones de Gabor, que vienen caracterizadas por una variación gaussiana en la amplitud, moduladas con un seno y escaladas y desplazadas en tiempo. Este diccionario de funciones es mucho más adecuado para representar sonido no estructurado que otros diccionarios como los que podrían ser un conjunto de funciones que varíen sólo en frecuencia (Fourier) o sólo en escalado en tiempo (Haar). Esto es debido a que Fourier está de alguna manera más dedicado a caracterizar señales de un rango de frecuencias superior, y Haar de un rango inferior, mientras que el sonido que pretendemos caracterizar es tan genérico que no sabemos a ciencia cierta a qué tipo se asemejará más.

Así pues, una función perteneciente a un diccionario de funciones de Gabor viene caracterizada por 4 parámetros, a saber s, u, ω y θ; correspondientes a la escala, el desplazamiento, la frecuencia y la fase respectivamente. Una función de Gabor genérica viene dada por la siguiente expresión:

$$g_{s,u,\omega,\theta}(t) = K_{s,u,\omega,\theta}\, g\left(\frac{t-u}{s}\right)\cos[2\pi\omega(t-u)+\theta]$$

con $g(t) = \frac{1}{\sqrt{s}}e^{-\pi t^2}$ y $K_{s,u,\omega,\theta}$ tal que $\left\|g_{s,u,\omega,\theta}\right\|^2 = 1$

El conjunto de parámetros s, u, ω y θ establece la posición de un átomo dentro del diccionario. En el caso concreto de nuestro proyecto, hemos elegido un diccionario donde todas las bases tienen $\theta = 0$, lo que simplifica bastante el tiempo computacional a la hora de generar el diccionario y los cálculos posteriores con éste. Hemos seguido una selección de parámetros condicionada por la elegida en la publicación donde se hizo el estudio del uso de funciones de Gabor como características de audio [3]. Para el resto de parámetros, y en el supuesto de que estemos trabajando con una ventana de longitud $N = 256$, hemos tomado los siguientes valores:

- Cuatro escalados, $s = [1,2,4,8]$, esto es, escalamos o diezmamos la función original para una función de Gabor genérica por cada uno de estos valores. Estos valores no varían con la longitud de la ventana.

- Cuatro desplazamientos $u = [0,64,128,192]$, esto es, desplazamos la señal original en tiempo para que tenga su máximo en cada uno de estos 4 puntos, dentro de los 256 iniciales. Estos valores sí varían con la longitud de ventana y se corresponden a un cuarto de la longitud, la mitad y tres cuartos.

- Un total de 30 frecuencias $\omega = i^{2.6}$, con $i = 1 \leq i \leq 30$, lo que nos da un conjunto de frecuencias que van distribuidas parabólicamente. Esta solución viene dada por el hecho de que necesitamos una mayor resolución a la hora de describir o modelar bajas frecuencias que altas, dado que según la experiencia de otros estudios relativos al audio no estructurado la zona de bajas frecuencias es donde más se va a marcar la diferencia entre unas clases de sonido u otros. Podemos ver la representación de éstas en la Figura 2. Estos valores no varían con la longitud de la ventana.

  Hemos decidido eliminar el rango de frecuencias que van en el rango de 7000 Hz a 10050 Hz (la mitad de la frecuencia de muestreo) por este mismo motivo, y porque el mero hecho de incluir 5 frecuencias más en nuestro diccionario, multiplicaba el tamaño de éste sin aportar datos de gran relevancia.

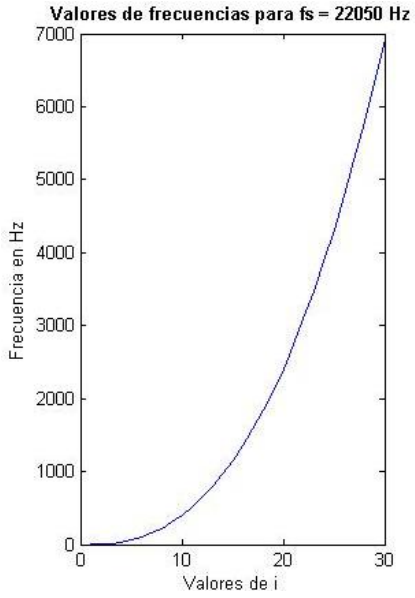

Figura 2 - Valores de frecuencias para el diccionario de funciones de Gabor

Contamos por tanto con un diccionario de 4*4*30 = 480 funciones de Gabor, lo que supone un costo computacional muy importante ya que en *matching pursuit* hace falta calcular la convolución de cada una de las funciones de la base con la ventana actual de audio. Dada la base de datos, de aproximadamente 20 minutos, contamos pues con un número elevado de ventanas, cada una de las cuales va a ser convolucionada 480 veces, y además cuanto más larga la ventana, más extensa la convolución. Esto va a resultar en un factor determinante en el transcurso de nuestro proyecto, dada la dificultad de la implementación de un sistema en tiempo real que sea capaz de realizar todos estos cálculos.

De hecho, este grave problema computacional ha supuesto que, a pesar de los estudios realizados durante el proyecto que confirman que la eficacia de las funciones de Gabor mejora con el incremento de la longitud de la ventana, dejemos de extraer los coeficientes de Gabor para longitudes de ventana mayores que 1024, reduciendo el número de características de las ventanas correspondientes a 2048 y 4096.

En la siguiente figura puede verse la reconstrucción de un fragmento de audio de 256 muestras como suma de 5 funciones de Gabor. Como hemos comentado anteriormente,

hemos decidido a limitarnos a un número más bien bajo de átomos para reconstruir la señal, ya que será la información de éstos primeros la más relevante a la hora de hacer clasificación. Además, la energía del error no se reduce demasiado a partir de éste número de átomos seleccionados.

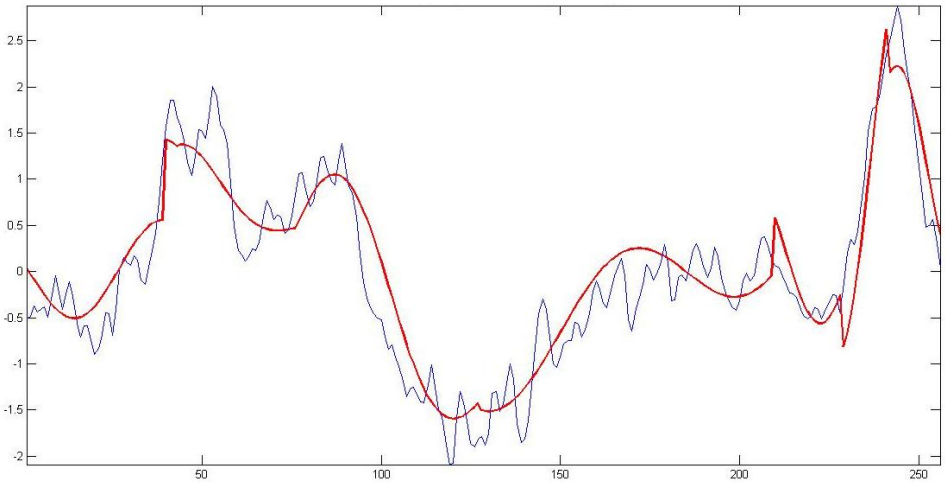

Figura 3 - Ventana de señal original y aproximación por matching pursuit con *n* = 5 átomos

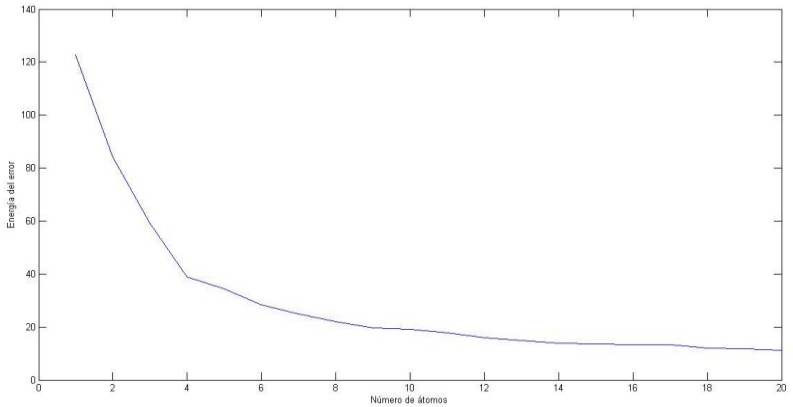

Figura 4 - Energía del error según el número de átomos escogidos para realizar *matching pursuit*

Una vez calculada la aproximación con 5 átomos a la ventana actual, hace falta extraer las características. Para esto se ha implementado un sistema de etiquetado que nos dice qué frecuencia, desplazamiento temporal y escalado se corresponden con cada átomo usado. La media y la desviación estándar de cada uno de estos parámetros constituye una pareja de características, por lo que obtenemos un total de 6 características a través de este método.

3.3.2 MFCC

Si hay alguna característica que se haya usado por excelencia en el reconocimiento de audio genérico ésta es los coeficientes cepstrales de las frecuencias de Mel. Los MFCC se han usado comúnmente en reconocimiento de habla, música y por supuesto también en sonidos ambientales y no estructurados. El hecho de usar una escala frecuencial basada en las frecuencias de Mel hace que esté enfocado hacia la percepción auditiva humana, ya que ésta escala de frecuencias, que consiste en una mera transformación matemática, fue diseñada para modelar de alguna manera el sistema perceptual auditivo humano.

En lo que a la extracción de características respecta, el algoritmo a seguir es muy claro. Una vez la señal ha sido enventanada (este es el único caso de características en el proyecto en el que la ventana empleada ha sido de Hamming), se realizan los siguientes pasos:

1. Para cada una de las ventanas, obtener el espectro en potencia.
2. Se aplica el banco de filtros de Mel al espectro en potencia.
3. Se suma la energía resultante del filtrado por cada uno de éstos.
4. Se toma el logaritmo de cada una de estas sumas
5. Por último se realiza la transformada discreta del coseno de estos logaritmos como si de una señal se tratara, obteniendo los MFCCs.

El banco de filtros de Mel es un conjunto de filtros triangulares en espectro cuyo espaciado entre sí se corresponde con el paso de la escala de Hertzios a la escala de Mel, mencionada anteriormente.

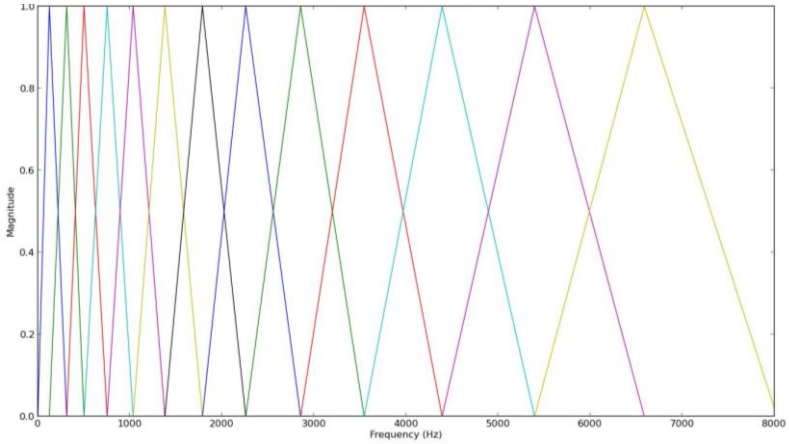

Figura 5 - Distribución de un banco de filtros de Mel sobre frecuencia en Hertzios

Estos coeficientes, de los cuales en nuestro proyecto hemos tomado 14 por recomendación de varias publicaciones, caracterizan estáticamente el trozo de señal que estamos evaluando en cada momento, lo cual hace que su eficacia varíe con la longitud del enventanado, dependiendo del tipo de señal de audio que estamos analizando. Además, como ya se ha mencionado en el capítulo 2, hay algunos estudios, mayoritariamente centrados en el estudio de música, donde se toma la primera y segunda derivada de los MFCCs como características para evaluar su velocidad y aceleración, esto es su avance temporal de una ventana a la siguiente.

3.3.3 Otras características

Basándonos en la experiencia de publicaciones sobre el audio no estructurado hemos decidido extraer otras características de cada ventana. Se trata de un conjunto de características de sencilla implementación, las cuales no tienen que ver demasiado unas con otras, aunque han sido implementadas dentro de la misma función para no hacer la transformada de Fourier cada vez que calculamos una de ellas. A continuación haremos una breve descripción sobre ellas:

- **Ratio de Cruces por Cero** (ZCR o *zero crossing rate*)

  Se trata de una característica que nos indica la cantidad de veces que se pasa de negativo a positivo o vice-versa dentro de una ventana. El valor va entre cero y uno,

siendo cero cuando toda la ventana se mantiene con el mismo signo y uno cuando el cambio de signo se produce siempre entre dos muestras consecutivas.

- **Ratio de energía por bandas** (BER o *band energy ratio*)

  Aunque no tiene una definición específica, el cálculo de unos BERs a otros varía muy poco. Se trata de calcular la FFT de la ventana actual y dividir su espectro en distintos segmentos, calculando la energía en cada uno de estos. El ratio puede ser cualquier cociente entre estas energías; en nuestro caso particular, hemos tomado la energía que hay en la segunda mitad del espectro (desde un cuarto hasta un medio de la frecuencia de muestreo) y la hemos dividido por la primera mitad (desde cero hasta un cuarto de la frecuencia de muestreo). Este ratio nos indica de alguna manera la relación que hay entre componentes de alta frecuencia y baja frecuencia.

- **Energía total**

  La energía total de una ventana también es susceptible de ser utilizada como característica. Se trata simplemente de la suma de los valores absolutos del vector obtenido al aplicar la FFT a la ventana original.

- ***Roll-off* en frecuencia**

  El *roll-off* en frecuencia determina dónde se encuentra la frecuencia que separa el 95% (o el porcentaje que sea) de la energía del 5% restante. Esto es da como resultado un valor de frecuencia para el cual se cumple que la energía desde esa frecuencia hasta la mitad de la frecuencia de muestreo es el 5% de la energía total de la señal.

- **Centroide espectral**

  El centroide espectral determina el área en frecuencia alrededor de la cual la mayor parte de la energía está distribuida. Está estrechamente relacionado con el brillo perceptual del sonido.

A continuación se muestra un ejemplo de estas características para un fragmento de audio de duración inferior a un segundo, el cual se divide en 121 ventanas de 256 muestras cada una.

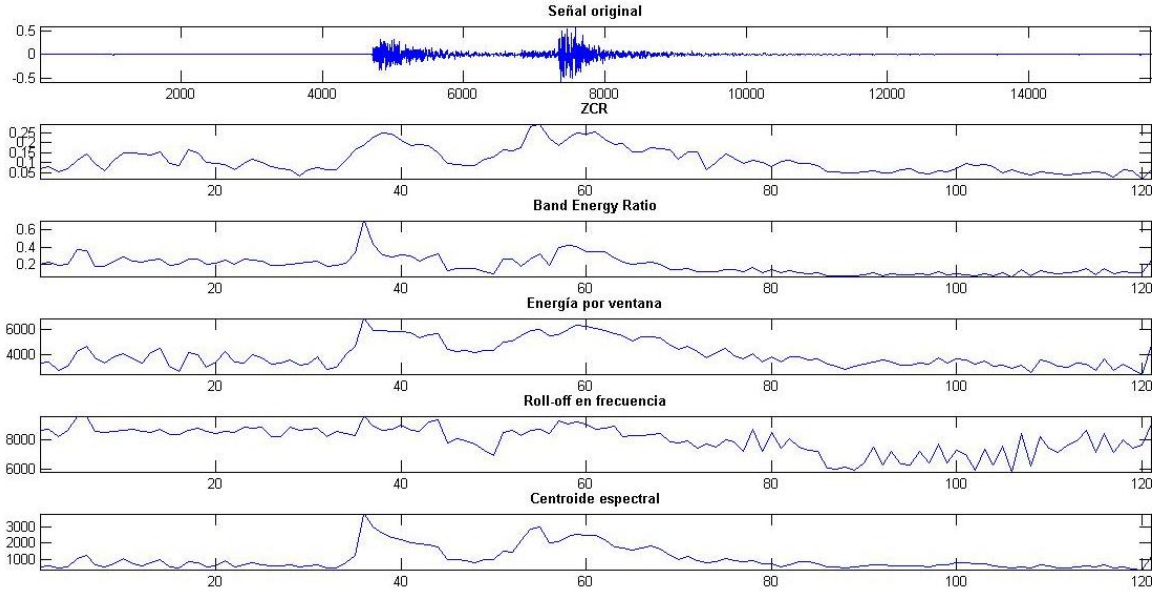

Figura 6 - Demostración de las otras características para un fregmento de audio corto

3.3.4 Vector de características

Posteriormente al cálculo de los 14 coeficientes correspondientes a MFCC, los 6 correspondientes a *matching pursuit*¸ y los 5 correspondientes a las características mencionadas anteriormente podemos crear un vector de 25 características correspondiente a cada ventana a través de la concatenación. Estas son las características con las que entrenaremos el mapa auto-organizado. Naturalmente cada vector lleva una etiqueta adjunta a éste en la cual se especifica la clase de sonido a la que pertenece, aunque esta no se usa a la hora de entrenar el mapa sino a la hora de comprobar lo bueno o malo que es mediante el cálculo del coeficiente Kappa.

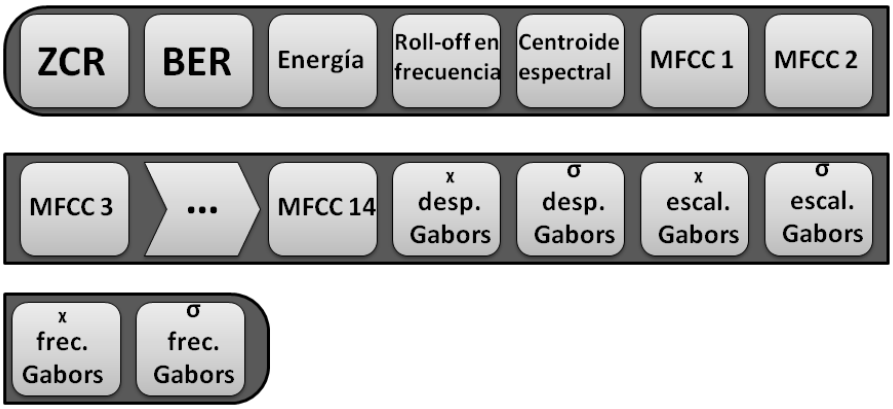

Figura 7 - Vector de características correspondiente a una ventana

Este vector es extraído de cada una de las ventanas de audio en las que se descompone la señal original, independientemente de la longitud de estas. Obviamente, la longitud de cada ventana influye notablemente en los valores obtenidos para cada una de las características, y de hecho la mejora del clasificador se va a basar en parte en hacer estudios sobre cuál es la longitud óptima de ventana.

3.4 Clasificación: Mapas auto-organizados

Los *self-organized maps*, SOM o mapas auto-organizados son un algoritmo de aprendizaje no supervisado especialmente apropiado para hacer análisis de datos, visualizaciones de datos multidimensionales, ordenaciones y *clustering*. Fueron desarrollados por Teuvo Kohonen en la *Helsinki University of Technology* [20].

SOM es un caso particular de redes neuronales artificiales basadas en un aprendizaje de modelo competitivo, esto es que las neuronas de la red compiten entre ellas para ser activadas. La entrada a la red consiste en todos nuestros datos extraídos anteriormente, es decir, una matriz de 25 variables de ancho, y con una longitud equivalente al número de ventanas en las que hemos dividido nuestra base de datos. La salida del algoritmo es un mapa formado por neuronas (la *SOM toolbox* calcula el tamaño del mapa automáticamente), donde cada neurona tiene un vector de pesos, en nuestro caso de 25 pesos ya que éstos son equivalentes al número de variables.

Al tratarse de un mapa que se extiende en 25 dimensiones del espacio usaremos una visualización específica para la representación del mapa. Es el caso de la U-matrix, donde la suma de las distancias euclídeas de una neurona a las vecinas viene representada por un tono de gris, de modo que una neurona viene representada por un hexágono de un color grisáceo rodeado de otras 6 neuronas. El nivel de oscuridad define la distancia, dentro del espacio de 25 dimensiones, a la que ese encuentra esta neurona de las demás, siendo una neurona casi negra una neurona que representa un sonido muy específico y muy diferenciado del resto por alguna de las 25 variables o una frontera entre *clusters* de neuronas. En contra, un grupo de neuronas blanquecinas representa una agrupación correspondiente a un tipo de sonido.

El algoritmo comienza inicializando las iteraciones y los pesos del mapa. El mapa "se extiende", esto es, reparte los pesos de las neuronas iniciales, en las dos direcciones de máxima variabilidad calculadas por PCA. Para ello se calculan los vectores propios de la base de datos, lo que para números de variables muy grandes puede suponer un problema computacional, lo que no es el caso. Posteriormente, el algoritmo es el siguiente [32]:

1. **Inicialización de los pesos** sinápticos de las unidades:
   Valores medios de los datos con pequeñas variaciones aleatorias.
   Patrones de entrenamiento escogidos en las dos direcciones de máxima variación PCA.

2. **Fase de recuerdo** con un patrón de entrada para encontrar la neurona ganadora a través de la mínima distancia euclídea. Dependiendo del factor de aprendizaje y del radio de vecindad habrá una cantidad mayor o menor de neuronas vecinas que se activen.

3. **Cálculo de la variación de pesos** de la neurona ganadora y sus vecinas:
   - Ajuste iterativo: las neuronas ganadoras ajustan sus pesos inmediatamente después del cálculo.
   - Ajuste cíclico: se acumulan las variaciones calculadas hasta haber presentado un ciclo de patrones.

4. **Finaliza el proceso de aprendizaje** en los casos:
    - Se alcanza el número máximo de iteraciones/ciclos establecido.
    - La variación de alguna función de monitorización se reduce por debajo de cierto umbral.

5. **En caso contrario, se vuelve al paso 2.**

Los ratios de aprendizaje y vecindad son los que van a hacer que aunque se active sólo una neurona (la de menor distancia euclídea con un vector de entrada dado), las de alrededor muevan sus vectores de peso hacia ésta, aunque cada vez en menor medida conforme el entrenamiento avanza. Finalmente, el algoritmo se detiene cuando la variación de los pesos de las neuronas se reduce considerablemente o cuando se alcanza un número límite de ciclos. El resultado final del mapa obtenido debería ser por tanto distintos *clusters* o agrupaciones de neuronas, cada una de ellas correspondiente a una clase de sonido.

A la hora de saber a qué sonido se corresponde cada neurona se calculan los llamados histogramas de activación, los cuales representan cuántas veces se activa cada neurona para cada tipo de sonido. Así pues, por poner un ejemplo, una neurona puede activarse 10 veces debido a los vectores de pesos de las ventanas correspondientes ruido de la calle y 100 veces debido a los vectores provenientes de el sonido de un coche. En este caso la neurona representa a los sonidos de tipo coche con mayor eficacia, y se le asigna esta etiqueta. Cabe la posibilidad de que una neurona no represente demasiado bien ninguna de las clases, y esto nos da la mayoría de fallos del clasificador. También existe la opción de que una neurona esté muy claramente entre dos clases sin pertenecer demasiado a ninguna de ellas: este es el caso de las neuronas interpoladoras, las cuales no llevan ninguna etiqueta asignada.

En un primer caso elegimos incorporar los datos tal cual los habíamos calculado en el proceso de extracción de características y obtener sus histogramas. En la figura 8 puede verse el resultado para 11 clases de sonido y 25 variables en ventanas de 256 muestras, con sus diferencias debidas a la longitud en ventanas de cada clase.

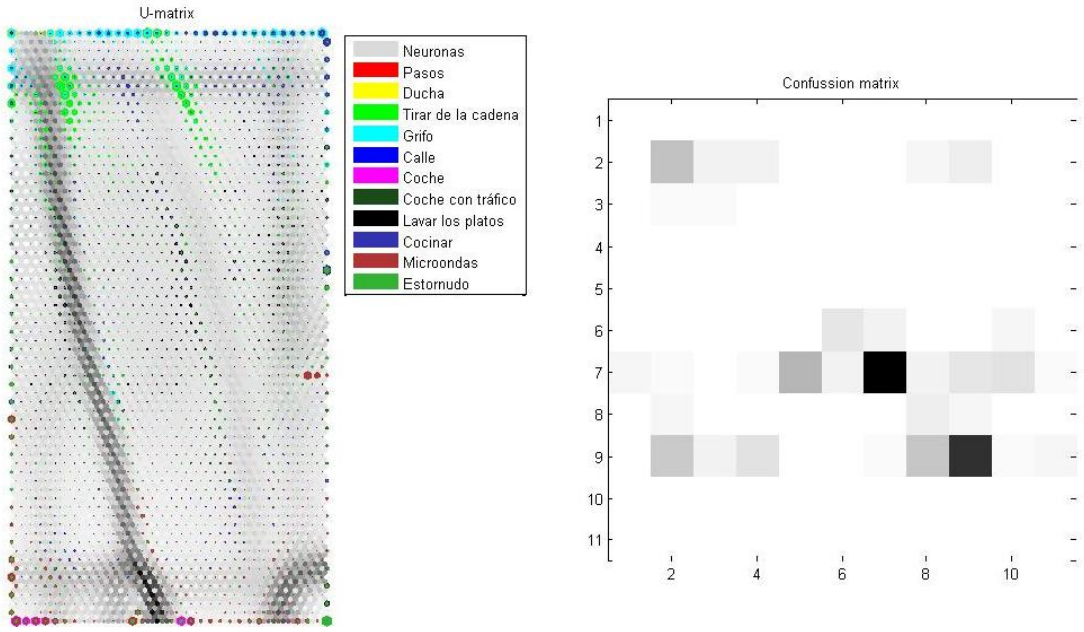

Figura 8 – SOM y matriz de confusión para 11 clases extraidas directamente de la base de datos y l.ventana = 256

Como puede apreciarse en la figura apenas ninguna clase forma un *cluster* de neuronas agrupadas entre sí. En todo caso el sonido correspondiente al coche ocupa la parte inferior del mapa y el sonido del retrete forma dos agrupaciones en la parte superior. El Kappa obtenido con este mapa es de 0.53, pero realmente si se ve la matriz de confusión de las clases se puede apreciar que es un mapa muy bueno para poder detectar el ruido producido en la cocina o en una calle con tráfico, precisamente las dos clases con mayor longitud de audio en la base de datos, pero no para detectar otras clases menos representadas.

De ahora en adelante, para tener una estimación de la calidad de nuestro clasificador nos basaremos en el coeficiente Kappa de Cohen y en la matriz de confusión. Siempre que mantengamos las mismas características de generación de mapas esta medida va a ser comparable, y por tanto vamos a tratar de maximizarla a lo largo de las siguientes secciones. Recuérdese que para que un mapa clasificador sea considerado de una calidad aceptable éste ha de superar un coeficiente de Kappa de 0.7 y la matriz de confusión ideal ha de ser similar a una matriz diagonal.

En el próximo capítulo nos dedicaremos a implementar una serie de mejoras que traten de aproximar tanto el coeficiente Kappa como la matriz de correlación a un resultado ideal.

Capítulo 4: Mejoras en el SOM, clasificador y simulaciones

En este capítulo vamos a presentar la serie de mejoras a través de las cuales hemos pasado de tener un SOM con una mala clasificación a un sistema un poco más complejo pero con unos índices de reconocimiento mucho mejores. Para ello primero hablaremos de las mejoras que se han podido hacer sobre el mapa original, posteriormente de la subdivisión del trabajo en varios mapas y por último del montaje y resultados del clasificador final, el cual ha sido simulado con dos señales de audio para evaluar su calidad.

4.1 Mejoras del clasificador manteniendo el SOM original

En este subapartado veremos cómo se ha seleccionado la base de datos óptima para trabajar con mapas auto-organizados y los diferentes estudios y análisis que se han hecho al respecto.

Nótese que todos los subapartados de esta sección han de ser leídos en su conjunto ya que dependen fuertemente unos de otros y se hacen referencias continuas entre ellos.

4.4.1 Equilibrio de vectores de pesos

Como se puede apreciar en la sección anterior, el hecho de tener una base de datos "desequilibrada" nos está empeorando la calidad del mapa. Con desequilibrada queremos decir que para algunos sonidos hay datos correspondientes al sonido de 3 o 4 minutos mientras que para otros hay sólo datos correspondientes a unas decenas de segundos. Esto afecta al mapa auto-organizado de manera que las clases de sonido que están más caracterizadas acaparan más neuronas, mientras que las menos caracterizadas pueden llegar a ocupar bastantes menos neuronas en relación. Esto hace que sea más difícil clasificar sonidos que están poco caracterizados ya que éstos tenderán a confundirse con las neuronas de los sonidos que son parecidos a ellos y estén más definidos.

Para solucionar este problema hemos implementado una solución muy sencilla, la cual consiste en equilibrar de alguna manera las longitudes del audio en la base de datos original. Escogemos una longitud '*lclase*' de vector de características intermedia entre las clases que menos audio tienen y las que más y hacemos lo siguiente:

- Para las clases con longitud del vector menor que *lclase*, se replica el vector de características mediante la función `repmat` de Matlab hasta llegar a la longitud de vector deseada. Así el mapa se entrena varias veces con los mismos vectores, lo que asegura la activación de las neuronas correspondientes más veces y un etiquetado más eficiente a posteriori.

- Para las clases con una longitud mayor, se hace un recorte, escogiendo *lclase* componentes aleatorias del vector, siempre que no se repitan. Al escoger componentes aleatorias aumenta la probabilidad de coger vectores de cada clip de sonido correspondiente a la clase, aunque en el caso de tener alguna característica que representase la evolución temporal de una ventana a la otra esta no sería una solución muy recomendable.

Esta solución va a ser además ligeramente condicionante, ya que hay clips de audio que representan mejor que otros las señales que están más caracterizadas, y los vectores que los caracterizan se eligen aleatoriamente al usar esta función. Así pues habrá longitudes de *lclase* para las cuales la base de datos sea óptima, e incluso el entrenar dos veces el mismo mapa en dos ejecuciones distintas del equilibrio de pesos nos puede dar Kappas distintos, precisamente debido a esta aleatoriedad.

La solución ideal (la cual será comentada en el capítulo 5: conclusiones y líneas futuras) consiste en tener una base de datos con la misma longitud de audio para cada una de las muestras, con lo cual evitamos este problema de la aleatoriedad y además cubrimos un rango mayor de clips, los cuales mejorarán seguramente la calidad de la clasificación.

Por otro lado, es conveniente remarcar que desde el momento en el que hacemos el equilibrio de longitudes estamos modificando las probabilidades a priori de las clases, distribuyéndolas unifromemente. El problema sigue siendo no-supervisado, pero ahora el clasificador estima la probabilidad a posteriori de cada clase. Implícitamente estamos modificando la base de datos para dar a entender que se tienen que formar 11 clases o *clusters*.

Los resultados de aplicar este equilibrio de pesos se observaron *ipso facto* en la matriz de clasificación (aunque el coeficiente Kappa disminuya a 0.3996), como se puede apreciar en la figura 9. En el caso de la matriz de confusión se ve cómo la diagonal se empieza a colorear de tonos oscuros (idealmente todos negros) aunque hay una gran cantidad de confusiones entre clases de sonidos. En el caso de la U-matrix podemos ver alguna agrupación más de neuronas que en el caso sin equilibrio de vectores de pesos.

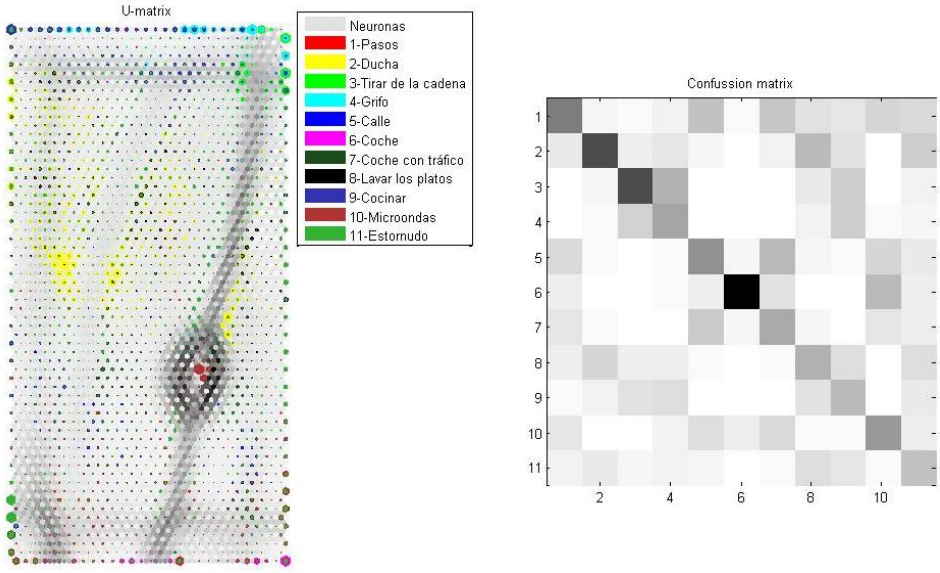

Figura 9 - SOM y matriz de confusión para 11 clases equilibradas con *lclase* = 12000 y l.ventana = 256 muestras

4.1.2 Eliminación de características mediante el uso del índice Kappa

Como ya se ha comentado varias veces a lo largo de la memoria, el uso de variables redundantes o que no aporten información relevante para la clasificación de las clases de nuestra base de datos puede ser un factor muy agravante a la hora de obtener un clasificador, dado que lo único que introducen es ruido.

Aunque nos hayamos basado en el estado del arte para hacer una selección de las variables que más nos interesan, necesitamos saber si estamos calculando características de más o si alguna de ellas es particularmente irrelevante para nuestro conjunto de datos. Ahora bien, el número de combinaciones de variables a escoger es de $2^{25} \cong 3.35 \cdot 10^7$ y no podemos plantearnos hacer todas las combinaciones posibles y calcular el coeficiente Kappa para cada una de ellas debido al excesivo coste computacional. Además, la relevancia de unas variables u otras puede depender de la longitud de la ventana de audio tomada, por lo que habría que hacer este cálculo para cada una de las longitudes de ventana estudiadas.

Hemos optado por dos algoritmos para calcular cuál es la mejor combinación de variables para cada ventana. Ambas se basan en que si una variable o conjunto de variables son relevantes para la clasificación, al suprimirlas el Kappa disminuye; y si no son relevantes, el Kappa aumenta.

- El **método manual** consiste en evaluar la calidad del mapa para todas las variables menos una para cada una de las variables, y posteriormente eliminar la que más haga crecer el índice Kappa, es decir, la que menos aporta a la clasificación. Una vez eliminada esta característica se evalúa otra vez la calidad, haciendo un bucle de eliminación de variables y evaluación de mapas con todas las variables restantes menos una, hasta que el Kappa deja de aumentar.
- El **algoritmo genético** es totalmente diferente. Se crea una población aleatoria de posibles combinaciones de variables, representadas como máscaras de unos y ceros. Se calculan los mapas y los correspondientes índices Kappa, para clasificarlas máscaras de mayor a menor efectividad, descartando una parte de éstas últimas y haciendo a las más eficientes "reproducirse" (crear nuevas máscaras con partes de las viejas) y en menor medida "mutar" (cambiar algún uno por cero o viceversa), con el fin de crear una nueva población-generación. Tras varias generaciones se puede conseguir un resultado óptimo, habiendo ahorrado el probar con las 2^{25} posiblidades.

La diferencia básica entre los dos algoritmos es que el algoritmo genético, siendo computacionalmente más lento y sin ofrecer el resultado óptimo con toda seguridad, tiene en cuenta las no linealidades derivadas del hecho de que dos o tres variables puedan "trabajar bien" juntas y no por separado, posibilidad que estamos descartando con el método de descarte de variables manual. No obstante en el caso específico de nuestro proyecto hemos obtenido mejores resultados con el método manual que con el algoritmo genético, el cual nos daba un Kappa inferior. Nótese que para que los índices Kappas sean comparables los tamaños de los mapas han de permanecer constantes independientemente de cuántas variables se hayan eliminado ya.

Como apunte muy importante, hemos visto para todas las longitudes de ventana estudiadas que la eficacia de las variables extraídas mediante *matching pursuit* está más que cuestionada. Esto nos ha llevado a hacer un pequeño estudio que se detalla en la sección 4.1.4. Este estudio, que contrasta con las publicaciones mencionadas en el estado del arte, nos ha llevado a abandonar el uso de estas 6 variables como parte del vector de características.

El número de características eliminadas para cada longitud de ventana se puede observar en la figura 10 del siguiente subapartado. No obstante, en todos los casos las 6 variables correspondientes a *matching pursuit* han sido eliminadas tarde o temprano.

4.1.3 Ventana óptima

Otro aspecto a mejorar durante el transcurso del proyecto ha sido el encontrar una longitud de ventana óptima para la cual el índice Kappa fuera máximo, de manera que con ella pudiéramos sacar el máximo partido de las características obtenidas. Para ello hemos extraído la base de datos para longitudes de ventana de 256, 512, 1024, 2048 y 4096 muestras, correspondientes a 11.6, 23.2, 46.4, 92.9 y 185.8 milisegundos respectivamente. Posteriormente hemos aplicado la eliminación de variables manual a cada una de ellas y hemos comparado los índices Kappa.

Por motivos reflejados en el apartado 4.1.4, para el caso específico de 2048 y 4096 muestras de longitud de ventana no hemos contado con las variables correspondientes a *matching pursuit*, por lo que la eliminación de variables ha empezado con 19 características por vector en lugar de 25.

Para hacer este experimento en igualdad de condiciones para todas las longitudes de ventana hace falta poner especial cuidado en el equilibrio de pesos mencionado en la sección 4.1.1: si *lclase* es igual a 12000 para una longitud de ventana de 256 muestras, lo lógico será que *lclase* sea la mitad para el doble de muestras por ventana, y así hasta tener *lclase* = 375 para 4096 muestras por ventana. También hay que prestar atención a conservar el tamaño del mapa sobre el cual se calcula el Kappa conforme se quitan variables y se alarga la ventana.

En la figura 10 se pueden ver los resultados correspondientes a la eliminación de variables para un mapa de tamaño constante de 27 x 12 neuronas, para los 5 tipos de ventana y para un equilibrio de pesos con *lclase*= 12000 en 256 muestras por ventana. Las líneas a rallas se corresponden con dos experimentos para las ventanas de 2048 y 4096 aumentando el tamaño del mapa a 36 x 13 neuronas. Esto lo hemos hecho así porque cada longitud de ventana (cada base de datos correspondiente) cuenta con un tamaño óptimo de mapa, el cual suele venir calculado automáticamente por la SOM *toolbox.* Se aprecia también que para este tipo de ventanas empezamos a eliminar variables desde la variable 19 y no la 25, debido a que para estas longitudes no contamos con las características provenientes de *matching pursuit*.

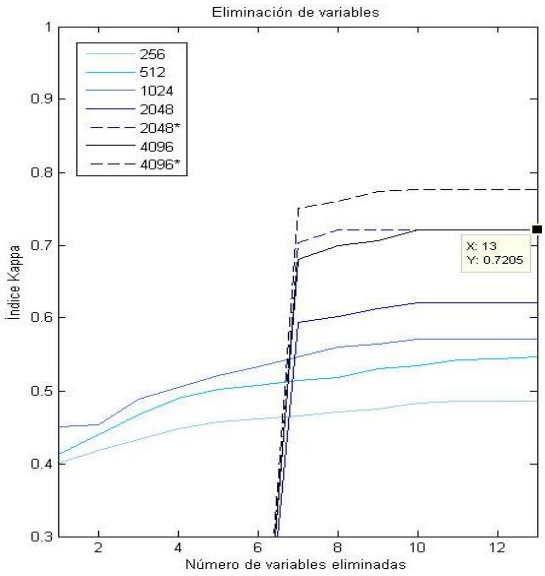

Figura 10 - Eliminación de variables para distintas longitudes de ventana

Como puede observarse en la figura la calidad de la clasificación y etiquetado obtenida con mapas auto-organizados se ve fuertemente incrementada con el aumento de la longitud de ventana, aún prescindiendo de las características correspondientes a *matching pursuit* en las dos ventanas más grandes, lo que da una nueva perspectiva a nuestro proyecto. De hecho para los 3 experimentos correspondientes a las longitudes de ventana de 256, 512 y 1024, estas 6 variables han sido eliminadas tarde o temprano, siendo la máscara de variables final una máscara sin variables *matching pursuit*.

A pesar de obtener un valor máximo de Kappa de 0.7759 para una ventana de 4096 muestras, hemos decidido tomar la de 2048 y buscar su tamaño de mapa y equilibrio de pesos óptimo, de manera que nuestro sistema no presente una latencia tan grande como la equivalente a 4096 muestras (185,8 milisegundos).

Después de decantarnos por una ventana de 2048 hemos realizado un estudio para saber el tamaño de mapa óptimo (el cual viene determinado por SOM *toolbox*) y el valor de *lclase* óptimo para esta longitud de ventana, una vez eliminadas las variables sobrantes, a saber las correspondientes a la frecuencia de Roll-Off y el centroide espectral. Recuérdese que en el apartado 4.1.1 se ha justificado por qué el tamaño de este parámetro puede influir en la calidad final del clasificador. Contamos pues al final con ventanas de 2048 muestras y 17 variables, 14 correspondientes a MFCC y 3 correspondientes al Coeficiente de Cruces por Cero, Ratio de Energía por Bandas y Energía total.

A continuación se presentan los resultados que nos ofrecen el valor de *lclase* óptimo, en función del coeficiente Kappa.

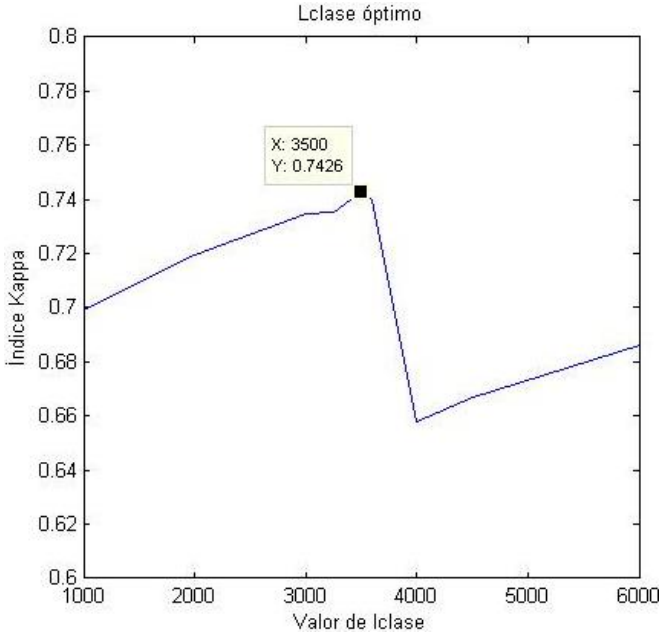

Figura 11 - Valor óptimo de *lclase* para una ventana de 2048 muestras

De aquí en adelante el proyecto se ha desarrollado con una ventana de 2048 muestras y 17 variables, equilibrada mediante un parámetro *lclase* igual a 3500 y un tamaño de mapa de 50 x 20. Nótese que hasta aquí seguimos trabajando con las 11 clases de sonido a la vez, sin haber evaluado matrices de confusión, y que más adelante estos cálculos han tenido que volver a ser hechos para otros tipos de base de datos que contienen menos clases.

4.1.4 Estudio de eficacia de *matching pursuit* como característica

Debido a la sorpresa que nos ha supuesto comprobar que las características provenientes de *matching pursuit* eran eliminadas tarde o temprano en los procesos de eliminación de variables, hemos decidido hacer un estudio sobre la eficacia a la hora de clasificar de estas características y por qué no han presentado el resultado esperado cuando se trata de caracterizar variables.

Para poder hacer este pequeño estudio hemos hecho algo tan sencillo como desarrollar mapas con distintas clases representadas solamente por las 6 variables correspondientes a *matching pursuit*, a saber la media y la desviación típica de la frecuencia desplazamiento y escalado. Esto no sólo lo hemos hecho para el caso de 11 clases sino también para el caso de 7 clases (pasos, retrete, calle, coche, platos, cocinar y microondas) y 3 clases (pasos, calle y microondas). Hemos mantenido también un tamaño de mapa de 49 x 37 neuronas y hemos ido reduciendo el tamaño de *lclase* mientras crecía el tamaño de ventana en muestras.

El hecho de reducir el número de clases viene fundado porque al tener que diferenciar entre menos tipos de sonido, un tamaño de mapa fijo se encargará de asignar más neuronas a cada tipo de clase, viéndose el Kappa incrementado. Estos son los resultados:

		Número de clases		
		11	**7**	**3**
Longitud de ventana	**256**	0.2481	0.3427	0.5007
	512	0.2849	0.3959	0.5943
	1024	0.3027	0.4694	0.7155

Tabla 2 - Índices Kappa para separación de clases solamente con variables *matching pursuit*

Las conclusiones de este estudio son bastante claras: estos 6 coeficientes son definitivamente una buena característica a la hora de clasificar ventanas de audio, y esta calidad mejora cuanto mayor es la longitud de la ventana empleada. Sin embargo, probablemente no sean suficientemente buenos combinados con las otras 19 características que hemos usado en el proyecto, ya que cuando se trata de separar entre 11 clases el índice Kappa es demasiado bajo.

El siguiente paso sería preguntarse el por qué no se han obtenido estas 6 variables para longitudes de ventana mayores que 1024 muestras, si se ve claramente cómo la calidad aumenta conforme aumenta la longitud de ventana. Esto será tratado con más detalle en el apartado 4.3.3 correspondiente a aspectos computacionales, pero podemos adelantar que la eficiencia computacional del cálculo de estas características es pésima y peor cuanto mayor es la longitud de la trama a caracterizar, haciendo una tarea difícil el cálculo de la base de datos para estas 6 variables en tramas de 2048 y 4096 muestras, e imposible una implementación en tiempo real.

4.2 Agrupamiento de clases y división en distintos SOM

Con la sección anterior hemos llegado a un punto en el que ya tenemos la longitud de ventana, mapa, variables y equilibrio óptimo para entrenar nuestro mapa auto-organizado pero, ¿es realmente una buena opción evaluar las 11 clases de golpe o es mejor agruparlas en conjuntos de sonidos para mejorar el reconocimiento? En este apartado responderemos a esta pregunta haciendo un estudio de lo que tenemos y construyendo un nuevo clasificador basado en cuatro SOM en un clasificador por etapas.

4.2.1 Estudio del agrupamiento de clases óptimo

Como hemos visto en apartados anteriores, al disminuir el número de clases entre las que hay que diferenciar la calidad del mapa mejora considerablemente. También hemos podido observar a través de las matrices de confusión que hay grupos de clases de sonido que tienden a confundirse más que otras. De estos dos hechos nace la idea de agrupar las clases que más se confunden en una misma clase, para que el SOM inicial se encargue sólo de descubrir que se trata de una super-clase, y posteriormente dedicar un segundo SOM para hacer exclusivamente la diferenciación entre estas clases, sin factores externos.

Para llevar esto a cabo ha hecho falta hacer un estudio sobre cuáles son las clases que más tienden a confundirse entre sí. Para ello no hay más que observar la matriz de confusión y el mapa correspondientes a una longitud de ventana de 2048, *lclase*= 3500, tamaño de mapa de 50 x 20 y 17 variables. Estos son los resultados, aparte de haber obtenido un índice Kappa de 0.7405:

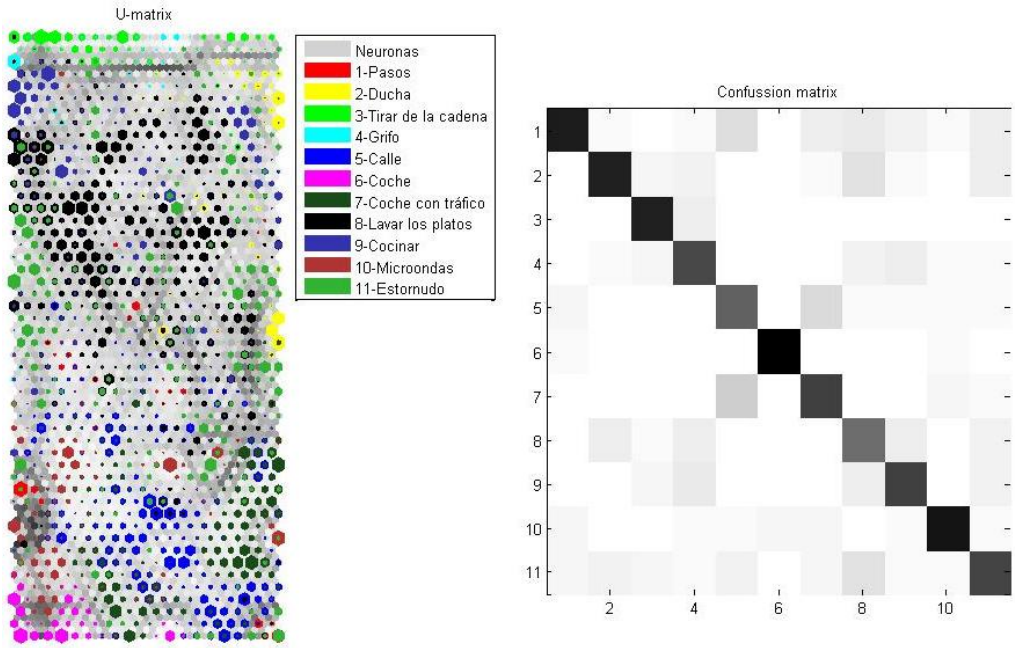

Figura 12 - Resultados tras hacer las mejoras correspondientes a la sección 4.1

Como puede verse en el mapa, hay varios sonidos que siguen muy esparcidos a lo largo de éste, aunque sin embargo ahora las neuronas están mucho más dedicadas a cada clase de sonido si las comparamos con figuras anteriores, siendo el etiquetado mucho más efectivo. En el mapa también se puede apreciar que clases como la 6 (rosa, coche), están muy agrupadas

en una zona del mapa, mientras que otras como la 8 (negra, lavar los platos) están muy esparcidas y a menudo confundidas con otras clases como la 11 (verde, estornudo) o la 2 (amarillo, ducha).

Estos conjuntos de confusiones se ven reflejados también en la matriz de confusión representada a su derecha. Así pues, la clase 6 no se confunde con apenas ninguna otra (es negra y no hay tonos grisáceos en la fila 6 ni columna 6), mientras que la clase 8 es gris oscuro y presenta confusiones con otras clases como la 2 y la 11 en sus respectivas fila y columna.

Tras evaluar e interpretar estas dos figuras hemos llegado a la conclusión que las clases que mejor se diferenciaban eran la 6 y la 10, y las peores la 8, la 11 y la 5. Una búsqueda de con qué clases se suelen confundir más frecuentemente nos ha llevado a hacer la siguiente agrupación de clases en grupos:

- En un primer nivel contaremos con un SOM que diferencia entre 3 clases y 3 super-clases, a saber “pasos”, “agua”, “fuera”, “coche”, “cocina” y “microondas”.
- En un segundo nivel contamos con 3 SOMs que diferenciarán las super-clases “agua”, “fuera” y “cocina” en las siguientes clases originales.
    - **Agua** contiene las clases “ducha”, “retrete” y “grifo”
    - **Fuera** contiene las clases “calle” y “calle con tráfico”
    - **Cocina** contiene las clases “platos”, “cocinar” y “estornudo”

El hecho de que este estudio nos haya proporcionado estos resultados fue una buena noticia en su día, ya que entra dentro de lo esperado que el reconocedor confunda entre los distintos sonidos correspondientes al agua o al estar en la calle. La única confusión poco esperada es la del sonido del estornudo con otros sonidos de la cocina como lavar la vajilla. Finalmente el diagrama de la parte correspondiente a SOM del clasificador tiene una estructura como la representada en la siguiente figura, de diagrama en árbol:

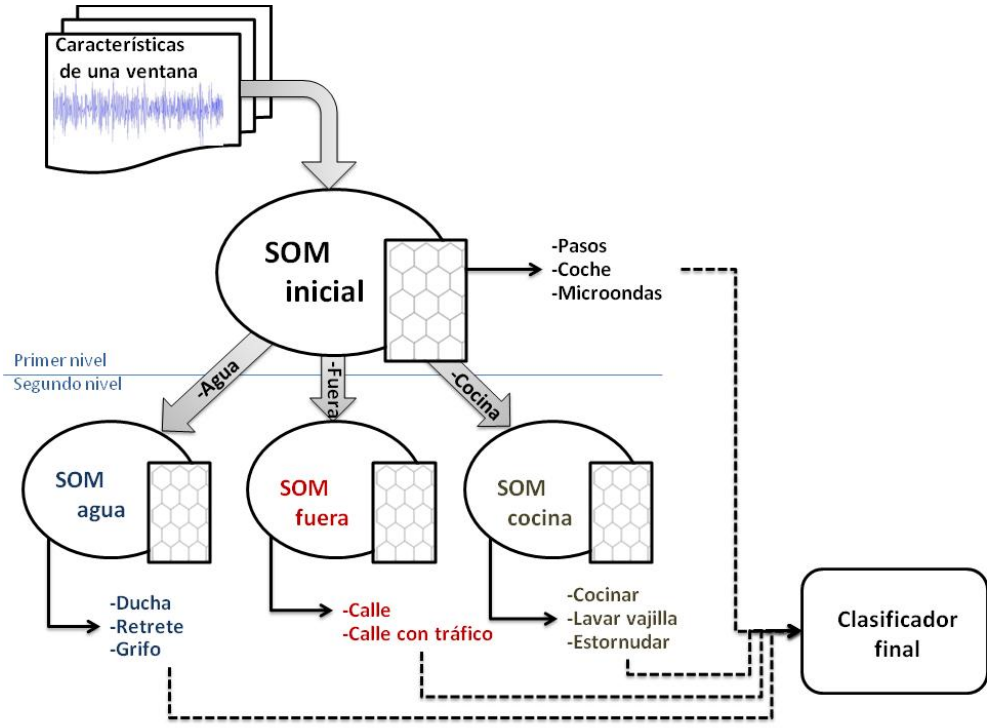

Figura 13 - Diagrama del funcionamiento de la parte del clasificador correspondiente a SOM

4.2.2 Implementación y resultados

Para implementar el código correspondiente a la parte anterior hace falta hacer una nueva estructuración tanto de la base de datos como de la estructura de mapas: para empezar la base de datos inicial ha de estar subdividida en las 6 clases (3 finales y 3 superclases) propuestas. Esto parece una tarea fácil a priori pero hemos de tener en cuenta que hay clases que cuentan con menor longitud que otras, y no queremos que un tipo de audio se vea infrarrepresentado dentro de una superclase, por lo que hay que hacer un equilibrio de vectores de pesos tanto en cada superclase como en el case de las 6 clases iniciales.

Usaremos el mismo algoritmo que para el ejemplo descrito en la sección 4.1.1 con la intención de saber el parámetro *lclase* y el tamaño de mapa óptimo para cada uno de los 4 mapas. Tras un estudio al respecto obtenemos un valor de *lclase* para el mapa de primer nivel de 6000, con un tamaño de mapa de 48 x 20 neuronas, obteniendo un índice Kappa de 0.8555. Como puede observarse en la siguiente figura, la subdivisión es muy efectiva, obteniendo matrices de confusión muy aproximadas a las ideales.

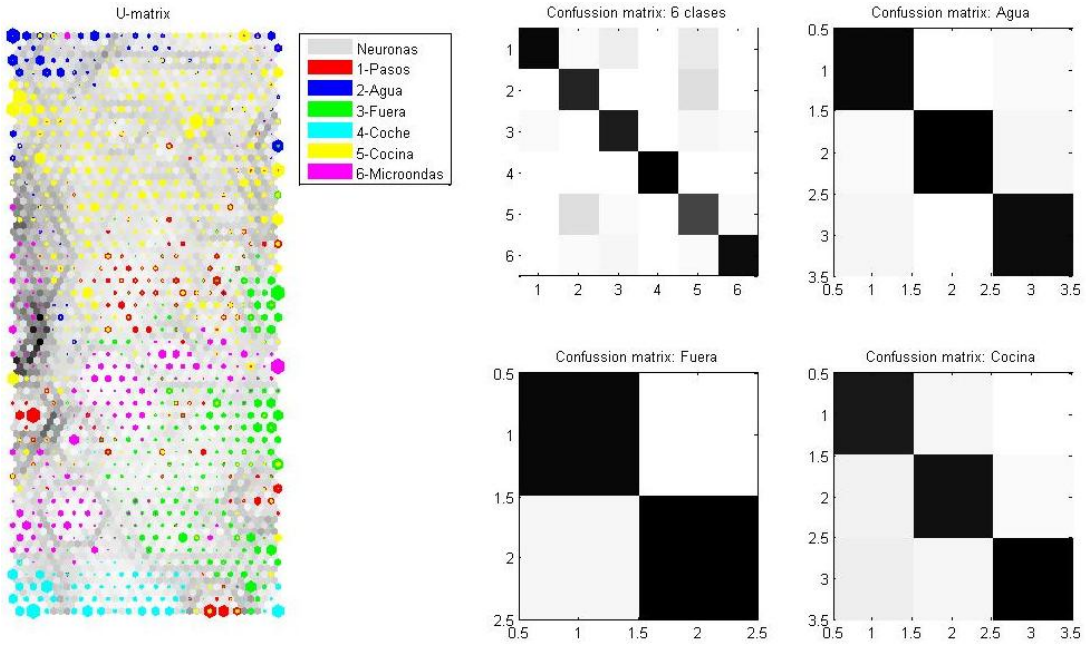

Figura 14 - SOM para 3 clases y 3 superclases, matriz de confusión correspondiente a éste y las 3 superclases

Los valores de la matriz de confusión para diferenciar 6 clases son los siguientes:

$$C = \begin{pmatrix} 5499 & 168 & 569 & 80 & 629 & 59 \\ 43 & 4976 & 57 & 8 & 782 & 83 \\ 218 & 41 & 4890 & 32 & 140 & 167 \\ 87 & 10 & 24 & 5811 & 33 & 15 \\ 109 & 706 & 166 & 18 & 4232 & 91 \\ 44 & 99 & 294 & 51 & 184 & 5585 \end{pmatrix}$$

Si hacemos una comparación con SOMs anteriores, podemos ver como en este la división en *clusters* proporciona mejores resultados, en parte debido a que ahora contamos con 6 clases en lugar de 11. En la siguiente figura se pueden ver los mapas correspondientes al segundo nivel, los cuales también presentan buenas agrupaciones de neuronas.

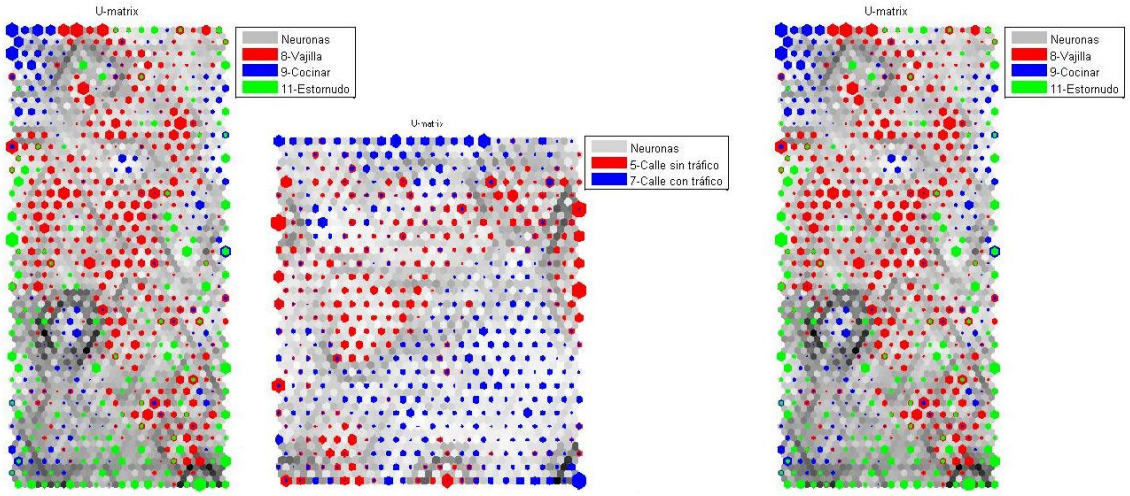

Figura 15 - SOMs correspondientes a las super-clases "agua", "fuera" y "cocina"

Los Kappas correspondientes a cada uno de los SOMs son 0.8611 para el que diferencia entre 6 clases; 0.9355 para el que diferencia entre sonidos provenientes del agua; 0.7836 para el que distingue sonidos de la calle y 0.8545 para el correspondiente a los sonidos de la cocina y estornudos: unos resultados muy aceptables y que nos dan pie a implementar dos simulaciones en tiempo real y el clasificador final. Por otro lado las matrices de confusión correspondientes a cada uno de estos mapas son:

$$C_{agua} = \begin{pmatrix} 5556 & 56 & 155 \\ 168 & 5803 & 259 \\ 276 & 141 & 5586 \end{pmatrix}, C_{fuera} = \begin{pmatrix} 4964 & 807 \\ 1036 & 5193 \end{pmatrix} \text{ y}$$

$$C_{cocina} = \begin{pmatrix} 4942 & 392 & 146 \\ 430 & 5116 & 219 \\ 628 & 492 & 5635 \end{pmatrix}$$

A partir de aquí hay que tener en cuenta que una vez que hemos realizado una subdivisión del problema como esta, el índice Kappa que nos ha hecho mejorar hasta este punto ya no tiene validez y tendremos que idear un nuevo coeficiente o índice para medir la calidad global del clasificador.

4.3 Clasificación de dos señales de sonido

En el último apartado de este capítulo vamos a explicar la estructura del clasificador final. Éste consiste en un simple filtro que transforma y suaviza la variabilidad de una serie temporal de etiquetas correspondientes a 185.2 milisegundos obteniendo unas etiquetas definitivas que limitan esta variabilidad.

También explicaremos el modo en el cual evaluamos la calidad del reconocedor final y los resultados para dos señales de audio, una de ellas construida artificialmente. Por último entraremos en los detalles computacionales del proyecto final.

4.3.1 Estructura del clasificador final y de las dos señales de sonido empleadas

A la hora de evaluar el funcionamiento de nuestro reconocedor hemos usado dos señales de audio creadas por nosotros. La primera de ellas ha sido obtenida a partir de la

misma base de datos que hemos usado para entrenar los mapas: simplemente nos hemos dedicado a unir con *Audacity* un conjunto de grabaciones independientes como si de diferentes pistas se tratara, haciendo que cada una comience cuando la anterior acaba. La segunda de ellas es una grabación casera realizada con el micrófono de un teléfono móvil Samsung Galaxy S Duo, posteriormente editada con *Audacity* para obtener una frecuencia de muestreo de 22050 Hz, que es la frecuencia a la que se ha desarrollado el proyecto.

Para cada una de ellas hemos creado dos "clasificaciones ideales", esto es, unas señales artificiales que nos indican lo que esperamos de las clasificaciones si estas fueran totalmente precisas. Hemos creado una secuencia de etiquetas ideal tanto para las 6 clases iniciales como para las 11 finales, es decir, dos por grabación.

A pesar de haber obtenido muy buena sensación con los índices Kappas de los 4 SOMs de los capítulos anteriores, es un hecho que sigue habiendo confusiones, lo que provoca que haya variabilidad de la etiqueta de una ventana a la siguiente, a pesar de que la mayoría de las etiquetas correspondientes a un fragmento de tiempo mayor se corresponden normalmente a la clase esperada. Podemos ver de lo que estamos hablando en la primera gráfica de la figura 16, representada en la siguiente sección.

Además hay que tener en cuenta que debemos de contar con un estado extra que no hemos entrenado, que sería el estado "silencio u otros sonidos", el cual asignaremos a cero. El motivo por el cual no ha sido entrenado es precisamente que no se puede modelar "todos los tipos de sonido que no sean una de las 11 clases" en una nueva clase, algo lógico dada la naturaleza de sonidos potencialmente grabables por un dispositivo móvil.

Para solventar estos pequeños problemas hemos recurrido a una solución tan sencilla como volver a hacer un enventanado temporal (esta vez de etiquetas en lugar de muestras) y calcular el histograma de cada ventana. Si el máximo valor del histograma no supera cierto valor de *threshold* , entonces en esta ventana temporal tenemos un estado cero, de silencio, transición de sonidos o cualquier otro sonido no reflejado en la base de datos. Sin embargo, si se supera este valor, la clase que participe en el máximo del histograma es aquella correspondiente a la longitud de toda la ventana. Este procesado es equivalente a calcular la moda entre las etiquetas de cada ventana temporal escogida, excepto por la parte correspondiente al *threshold*.

Sea n_i la frecuencia absoluta de la clase i en una ventana de audio de N etiquetas , $f_i = \frac{n_i}{N}$ la frecuencia relativa correspondiente, y sea un *threshold* θ :

$$\text{Sea } f_k = max(f_i) \begin{cases} si\ f_k < \theta \Rightarrow etiqueta = 0 \\ si\ f_k \geq \theta \Rightarrow etiqueta = k \end{cases}$$

Para medir la calidad de nuestro reconocedor final hemos optado por crear un índice que va entre cero y uno, explicando el error relativo entre las etiquetas obtenidas y las ideales. Para ello simplemente se hace un cociente entre el número de aciertos (etiqueta obtenida igual a la ideal) y la longitud total de la señal de audio en ventanas. Esta medida de la calidad se ha aplicado tanto para 6 clases como para 11 clases con intención de evaluar los resultados.

4.3.2 Resultados

A la hora de evaluar la calidad del clasificador final nos hemos centrado en el índice explicado anteriormente, el cual nos da una estimación de la relación entre el etiquetado esperado o ideal y el obtenido realmente mediante el método basado en SOM. También viendo las gráficas puede uno hacerse una idea del nivel de acierto en la estimación de cada tipo de sonido.

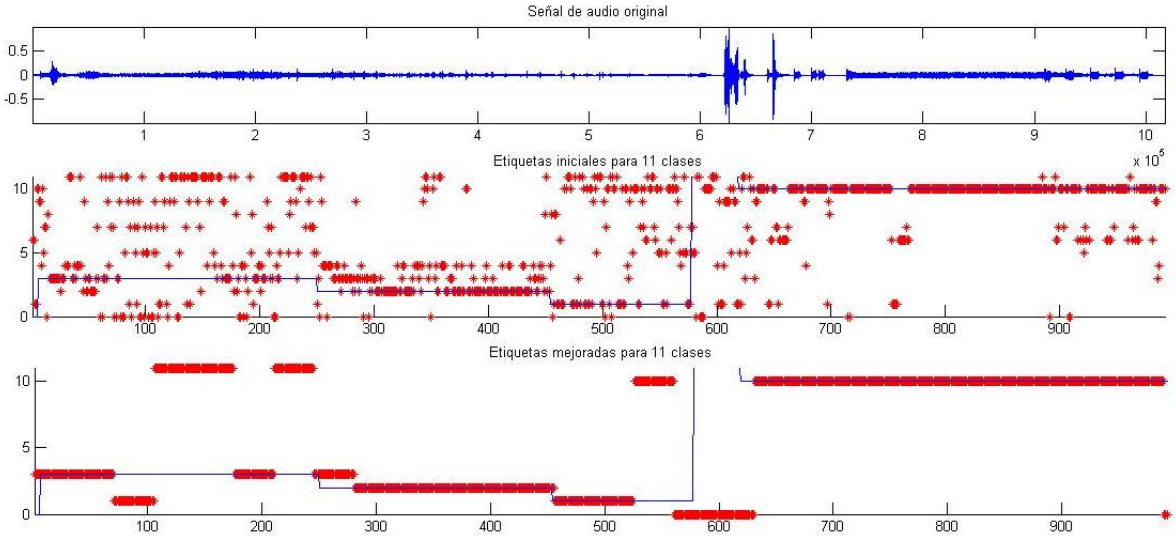

Figura 16 - Etiquetas iniciales y post-procesado para la grabación artificial hecha a partir de la base de datos

Para el caso de la señal de audio construida artificialmente a partir de sonidos de la base de datos obtenemos índices de calidad entre 0.5 y 0.75 para la clasificación de 11 clases por norma general, dependiendo este valor en pequeña medida de cómo ha sido generada la base de datos (especialmente de la parte aleatoria de ésta, correspondiente al equilibrio de vectores, que se genera cada vez de una manera distinta). Podemos ver cómo sonidos como el 2 (ducha) y el 10 (microondas) se reconocen con casi total precisión, mientras que alguno como el 11 (estornudo) tiende a confundirse a menudo con otros sonidos.

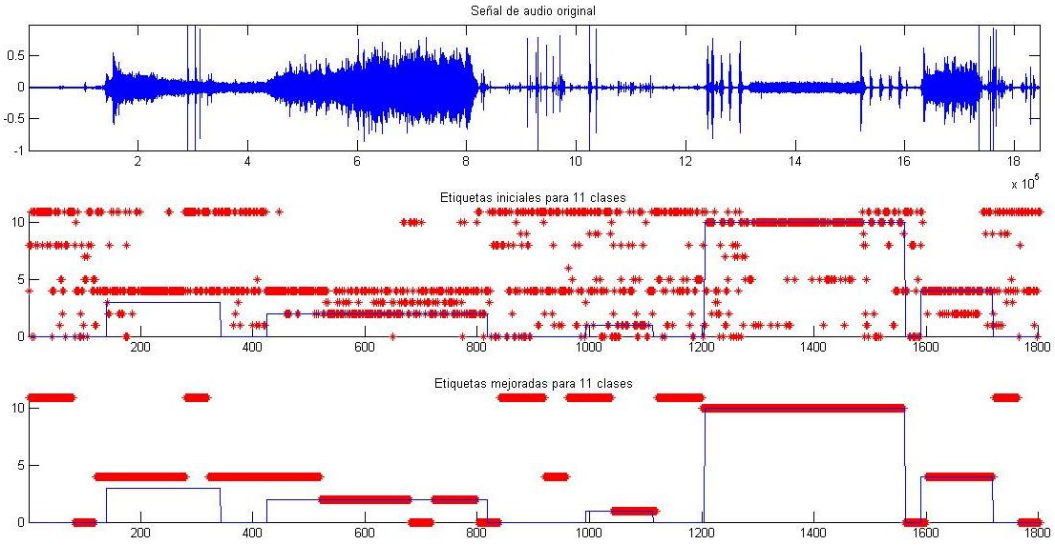

Figura 17 - Etiquetas iniciales y post-procesado para grabación hecha con el móvil

Para el caso de la señal real grabada con un teléfono móvil, los índices de calidad del reconocimiento suelen variar entre 0.4 y 0.6. Volvemos a contar con el problema del sonido número 11, que tiende a confundirse con otros más a menudo de lo que nos gustaría. En este caso el microondas (clase 10) y el sonido de un grifo (número 4) se reconocen perfectamente, mientras que el sonido de la ducha (2) se reconoce parcialmente, y el del retrete (3) directamente no se reconoce en absoluto.

Estos resultados varían en su índice de calidad de un equilibrio de vectores al siguiente, por lo que en el apartado de conclusiones y líneas futuras se hablará de cuáles son las ventajas y desventajas de hacer este equilibrio.

4.3.3 Aspectos computacionales

En el último apartado de este capítulo comentaremos brevemente los aspectos computacionales del proyecto que se ha llevado a cabo. La integridad del proyecto se ha llevado a cabo con un procesador Intel Core Duo T600 a 2.2 GHz.

Para empezar vamos a aclarar cuál es el problema que ha desechado el cálculo de funciones de Gabor como base en *matching pursuit* para ventanas de más de 1024 muestras. Para saber cuáles son las funciones de la base (recordemos que la base cuenta con 480 funciones) que mejor caracterizan a una trama de audio, hace falta hacer la convolución de la trama con cada una de estas 480 funciones, almacenarlas en una matriz C y localizar los máximos.

Además, con el fin de obtener las 5 funciones que convenientemente ponderadas nos dan la mejor aproximación a la trama, hace falta actualizar la matriz haciendo referencia a una matriz P que contiene la convolución de cada función de la base con sí misma y todas las demás. Las dimensiones de esta matriz tridimensional son 480 x (tamaño de ventana x 2 -1) x 480, teniendo un tamaño mayor cuanto mayor es el tamaño de ventana y llegando a ocupar 2 Gb (el máximo permitido por Matlab para una variable) para el caso de 1024 muestras. La referencia a esta matriz es indispensable dado que hace falta actualizar la matriz C para eliminar la posibilidad de que una misma función de la base represente dos veces a la trama de audio. El almacenamiento y las excesivas convoluciones correspondientes a esta función provocan que no sea factible la implementación de *matching pursuit* para tamaños de ventana mayores a 1024.

En el caso de las MFCCs y las demás características, todas ellas, excepto el ratio de cruces por cero, hacen uso tarde o temprano de la Transformada Rápida de Fourier. En el caso del centroide espectral, el ratio de energía por bandas, la energía y el roll-off en frecuencia esta FFT se aplica directamente tras pasar la señal de audio por la ventana rectangular, por lo que éstas han sido agrupadas en una sola función para aprovechar la misma transformada en todas ellas. En el caso de las MFCCs la FFT se aplica tras pasar la señal de audio por una ventana tipo *Hamming*, por lo que hay que hacerla a parte; y además se hace una transformada discreta del coseno, la cual tiene el mismo coste computacional que las FFT.

En la siguiente figura puede verse el tiempo de cálculo de extracción de cada una de las características correspondientes a una ventana de audio. La extracción de “otras características” implica la suma de todas las que no son *matching pursuit* o MFFCs. La medida de los tiempos se ha hecho con las funciones `tic` y `toc` de Matlab, para una misma ventana.

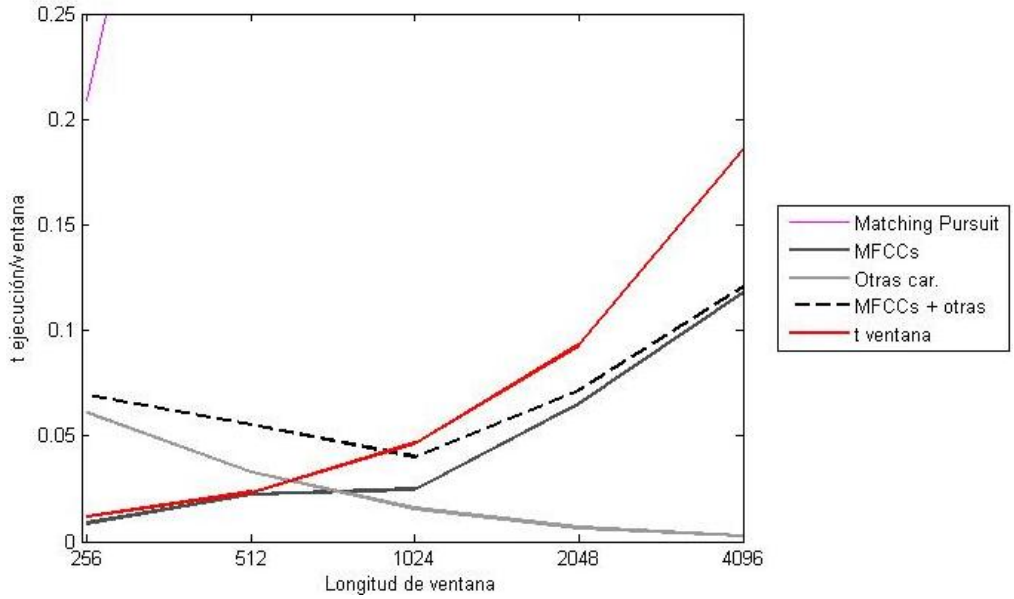

Figura 18 - Tiempo de extracción de características para distintas longitudes de ventana.

Nótese que los tiempos de cálculo de *matching pursuit* se salen ya de la gráfica para una ventana de 512, de hecho no los hemos incluido porque si no el resto de tiempos eran inapreciables en la gráfica.

Hemos incluido la suma de tiempos de MFCCs con las otras características para compararlos con el tiempo equivalente a cada ventana (en rojo). En el momento esta suma de tiempos de cálculo de características es menor que el tiempo real que ocupa la ventana, éstas características pueden ser calculadas en tiempo real para ser introducidas en un buffer y proceder a la búsqueda de la neurona más parecida en la tabla.

Capítulo 5: Conclusiones y líneas futuras

En este último capítulo vamos a comentar las principales conclusiones que hemos sacado del proyecto, las mejoras factibles a la hora de mejorarlo y las líneas en las cuales creemos que se puede hacer cierta investigación futura para acabar teniendo un sistema empotrado que ayude a personas dependientes.

Es difícil hacer una diferenciación entre mejoras y líneas de trabajo futuras ya que van muy ligadas unas con otras, por lo que hemos presentado las mejoras más a corto plazo en la primera sección y las líneas de investigación más a medio-largo plazo en la segunda.

5.1 Conclusiones y posibles mejoras del proyecto

Estas son en general las conclusiones que hemos tomado de los resultados obtenidos del proyecto y las mejoras propuestas a un plazo más corto de tiempo.

La primera conclusión evidente es que sí se puede hacer un sistema clasificador de sonidos no estructurados basado en redes neuronales y que además puede dar muy buenos resultados.

Para empezar, en la sección de la creación de la base de datos creemos que el hecho de que las longitudes correspondientes a cada clase de audio estén descompensadas ha influido negativamente en el proyecto, ofreciendo resultados variables en distintas ejecuciones y teniendo que crear una función que se encargue de compensarlas artificialmente. Hacer un equilibrio de longitudes no sólo ha implicado tener que implementar la función que equilibra sino también la búsqueda de una longitud óptima para cada agrupamiento de clases. Para otras ocasiones o líneas futuras convendría realizar una base de datos que contenga la misma longitud de audio de cada clase, lo que puede conllevar el tener que hacer grabaciones, dado que la base de datos de *freesound.org* no es mucho más extensa de lo que hemos usado en este proyecto, aunque bien es cierto que crece constantemente.

En cuanto a la extracción de características nos hemos reafirmado en cuanto a que MFCCs es una buena caracterización del audio en general: estas características junto con otras 3 han sido suficientes para hacer una caracterización de clases de sonido con buenos resultados. Por otro lado, hemos rebatido la idea defendida en algunas publicaciones como [3] de que los MFCCs funcionan mejor combinados con *matching pursuit* que por sí solos, además de probar que *matching pursuit* es muy ineficiente computacionalmente. Eso no implica que los parámetros extraídos a través de este método aporten nada a la hora de clasificar sonido, como hemos reflejado en nuestro estudio de la sección 4.1.4.

A la hora de extraer etiquetas hemos obtenido unos resultados más variables de lo que nos hubiera gustado, lo que nos ha llevado a implementar la subdivisión de un solo mapa en 4 mapas y el agrupamiento de clases en super-clases. Estos resultados serán tanto mejores cuanto más estudio se hace sobre la base de datos y las matrices de confusión con las que se está trabajando: una conclusión que se deduce de esto mismo es que en el caso particular de redes neuronales la calidad de la clasificación depende enormemente de la base de datos y el

tipo de clases que se quieren reconocer, teniendo que hacer un cálculo y análisis de las matrices de confusión para encontrar una subdivisión óptima del problema.

En lo que se refiere al clasificador final, el cual en nuestro caso hace un cálculo de la moda de las etiquetas por ventana temporal, creemos que este sistema, aún siendo eficiente, puede ser mejorado mediante modelos ocultos de markov, ya que así se pueden modelar ciertas características. Por ejemplo, es muy poco probable o imposible que suene un microondas mientras se está en la calle, o una ducha a la vez que el sonido del coche. Este tipo de incongruencias se pueden reflejar en las probabilidades de transición de un estado a otro en HMMs. El estudio de estas probabilidades también nos podría ayudar para hacer una limpieza de *outliers* en la base de datos en caso de que fuera necesario.

En la misma línea, también existe la opción de modelar de alguna manera la ocurrencia de dos o más clases de sonido a la vez. Un ejemplo de ello podría ser el andar mientras se está en la calle o el estornudar, un evento que ocurre generalmente mientras se hacen otras tareas.

5.2 Líneas futuras

Aquí vamos a presentar las que consideramos pueden ser las líneas que lleven a este reconocedor a convertirse en un producto final embebido.

Para empezar habría que hacer un estudio e implementación de todas las mejoras propuestas en la sección anterior para ver hasta qué punto mejora el sistema en su conjunto. Una posible manera de aumentar la base de datos es la inclusión de alguna manera del propio audio que se está clasificando como parte de la base de datos: además de dedicar el audio grabado para ser clasificado lo podemos usar para entrenar los SOMs a posteriori. Los inconvenientes que le vemos a esta solución es un desequilibrio de las longitudes de audio para cada clase de nuevo y el tener que confirmar mediante algún interfaz que se está haciendo un sonido de una clase y no de otra. Además un aumento excesivo del trabajo de procesado podría obligarnos a implementar parte del trabajo *offline*¸ almacenando la base de datos o los mapas en la nube (*cloud computing*).

Una posibilidad de cara a obtener mejores resultados en la clasificación es la inclusión de variables no provenientes del sonido sino de algún otro tipo de sensor como ya se ha comentado en la sección del estado del arte. Añadir sensores como infrarrojos o acelerómetros en ciertos puntos estratégicos puede incrementar la calidad enormemente, aunque lleva el estudio a un nuevo nivel del cual el sonido es sólo una parte del proyecto final.

En aras de llevar el sistema a una implementación final es necesario portar el código a algún lenguaje de programación que soporte la entrada de audio y la salida de datos en tiempo real. Al fin y al cabo el sistema final constará de un microcontrolador o microprocesador que se encargue de recibir interrupciones con la entrada de datos de audio para ser procesados en tiempo real y acabar asignándoles una etiqueta.

También creemos que sería necesario, de cara a una implementación final del producto, la charla con un profesional terapeuta ocupacional o médico. Con una ayuda de este tipo se puede hacer un estudio sobre cuáles son las secuencias de sonidos-actividades que entran dentro de la normalidad y cuáles deberían llevar a nuestro sistema a crear una alarma:

la falta de pasos o sonido del cuarto de baño durante varios días, la estancia en la calle por más de cierto número de horas, un aumento excesivo del número de estornudos o el paro repentino del sonido de un coche seguido del sonido de gente gritando o ambulancias, por poner varios ejemplos. De hecho, podría programarse de alguna manera para que el sistema no sólo informe a la persona correspondiente, sino que también actúe en consecuencia, por ejemplo haciendo una búsqueda de la farmacia abierta más cercana o enviando un mensaje al un teléfono de emergencias como el 112.

Tras el estudio e implementación de todos o alguno de estos pasos podemos evaluar resultados e implementar el sistema en un dispositivo portable, con el fin de que las personas con dependencia y las que se encargan de ellas tengan una mejor comunicación a distancia.

Bibliografía

[1] Celler, Branko G., Nigel H. Lovell, and D. K. Chan. "The potential impact of home telecare on clinical practice." *The medical journal of Australia 171.10* (1999): 518-521.

[2] de España, Proyección de la Población. "a Corto Plazo 2013-2023. Instituto Nacional de Estadística 2013. Nota de prensa.

[3] Chu, Selina, Shrikanth Narayanan, and C-CJ Kuo. "Environmental sound recognition using MP-based features." *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008.

[4] http://audacity.sourceforge.net/

[5] http://es.mathworks.com/products/matlab

[6] http://www.cis.hut.fi/somtoolbox/

[7] Schilit, Bill, Norman Adams, and Roy Want. "Context-aware computing applications." *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. IEEE, 1994.

[8] Chu, Selina, et al. "Where am I? Scene recognition for mobile robots using audio features." *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2006.

[9] Malkin, Robert G., and Alex Waibel. "Classifying user environment for mobile applications using linear autoencoding of ambient audio." *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*. Vol. 5. IEEE, 2005.

[10] Vacher, Michel, et al. "Complete sound and speech recognition system for health smart homes: application to the recognition of activities of daily living."*New Developments in Biomedical Engineering* (2010): pp-645.

[11] Katz, Sidney, and C. A. Akpom. "12. Index of Activity of Daily Living." *Medical care* 14.5 (1976): 116-118.

[12] http://creativecommons.org/publicdomain/

[13] https://www.freesound.org/

[14] Davis, Steven, and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences." *Acoustics, Speech and Signal Processing, IEEE Transactions on*28.4 (1980): 357-366.

[15] Zhu, Qifeng, and Abeer Alwan. "On the use of variable frame rate analysis in speech recognition." *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. Vol. 3. IEEE, 2000.

[16] Loughran, Róisín, et al. "The use of mel-frequency cepstral coefficients in musical instrument identification." *International Computer Music Conference, Belfast, Northern Ireland*. 2008.

[17] Dörfler, Monika. "Time-frequency analysis for music signals: A mathematical approach." *Journal of New Music Research* 30.1 (2001): 3-12.

[18] Wang, Avery. "An Industrial Strength Audio Search Algorithm." *ISMIR*. 2003.

[19] Sehili, M. A., et al. "Daily sound recognition using a combination of gmm and svm for home automation." *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012.

[20] Kohonen, Teuvo. "Self-organized formation of topologically correct feature maps." *Biological cybernetics* 43.1 (1982): 59-69.

[21] Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." *Signal Processing Magazine, IEEE* 29.6 (2012): 82-97.

[22] Kiani, Kourosh, Chris J. Snijders, and Edzard S. Gelsema. "Recognition of daily life motor activity classes using an artificial neural network." *Archives of physical medicine and rehabilitation* 79.2 (1998): 147-154.

[23] Salman, A. H., et al. "Review of digital heart sound classification methods via Artificial Neural Networks." *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2013 3rd International Conference on*. IEEE, 2013.

[24] Matsui, Tomoko, and Sadaoki Furui. "Comparison of text-independent speaker recognition methods using VQ-distortion and discrete/continuous HMM's."*Speech and Audio Processing, IEEE Transactions on* 2.3 (1994): 456-459.

[25] Sehili, M. A., et al. "Daily sound recognition using a combination of gmm and svm for home automation." *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012.

[26] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20.3 (1995): 273-297.

[27] Chechik, Gal, et al. "Large-scale content-based audio retrieval from text queries." *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. ACM, 2008.

[28] Guo, Guodong, and Stan Z. Li. "Content-based audio classification and retrieval by support vector machines." *Neural Networks, IEEE Transactions on* 14.1 (2003): 209-215.

[29] Cohen, J. "A coefficient of agreement for nominal scales. E&c. Psychof."*Measuremenf 20* 37 (1960).

[30] Holland, John H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975.

[31] Lee, Chang-Hsing, Chin-Chuan Han, and Ching-Chien Chuang. "Automatic classification of bird species from their sounds using two-dimensional cepstral coefficients." *Audio, Speech, and Language Processing, IEEE Transactions on*16.8 (2008): 1541-1550.

[32] Buldain, David J. "*Neuro_SOM*" Apuntes de la asignatura de Neurocomputación.