Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño e implementación de un sistema de detección de patologías en la voz utilizando aprendizaje automático.

Design and implementation of a voice pathology detection system using machine learning.

Autor/es

Álvaro Gimeno Sinués

Director/es

Eduardo Lleida Solano

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Especialidad: Sistemas de Telecomunicación

2021

RESUMEN

Este trabajo de fin de Grado aborda la problemática del uso de técnicas de aprendizaje automático en tareas con pocos recursos de datos como es la detección de patologías.

El principal objetivo del proyecto es diseñar e implementar un sistema que sea capaz de detectar la presencia de patologías en la voz utilizando técnicas avanzadas de aprendizaje automático.

Se han utilizado dos bases de datos diferentes. Una, la más pequeña (VOICED), con 208 voces en total, divididas en 73 sanas y 135 patológicas. La otra base de datos (Saarbrücker), la más grande, tiene más de 2000 voces, divididas por 687 sanas y 1356 patológicas.

De las dos bases comentadas en el párrafo anterior, se han extraído una serie de parámetros para poder implementar el sistema de detección de patologías: parámetros frecuenciales como los coeficientes MFCC (Mel Frequency Cepstrum Coefficients), parámetros glotales, parámetros de entonación y diversos parámetros de audio con un conjunto de herramientas de código abierto llamadas OpenSMILE.

A partir de los parámetros comentados, hacemos uso de redes neuronales, modelos de mezcla gaussianas (GMM) y un clasificador SVM (Support Vector Machine) para llevar a cabo el aprendizaje automático. He de recalcar que, para poder reducir la dimensionalidad de diversos vectores de entrada a estos clasificadores, se usa una técnica basada en supervectores, donde se recoge la misma información, pero de forma más compacta para una mayor eficiencia.

Para finalizar veremos ciertos resultados de varias pruebas realizadas con los clasificadores anteriores, donde se podrá ver que, con la ayuda de los supervectores, llegamos a aumentar el rendimiento de los clasificadores. Hay que destacar que con los parámetros obtenidos con el conjunto de herramientas de OpenSMILE se obtiene una gran eficiencia con los tres tipos de clasificadores.

AGRADECIMIENTOS

A mi tutor Eduardo Lleida por su asesoramiento y paciencia en este trabajo.

A mi madre y a mi padre que siempre están ahí.

A mis amigos de siempre, con una mención especial a David, por ser como un hermano para mí y por aguantarme cada día, a pesar de la distancia.

A mis compañeros de clase que he conocido estos años, que gracias a ellos se han hecho más amenos.

ÍNDICE

1. INTRODUCIÓN

1.1. Contexto

La tecnología cada vez está más presente en nuestras vidas, ya que está evolucionando a unos niveles desmesurados. La inteligencia artificial cada vez es más utilizada, por ejemplo, hoy en día, numerosas familias tienen instalados asistentes virtuales en sus casas (como Alexa) o en sus dispositivos móviles (como Siri) para recordar ciertas actividades que tenemos que realizar, reproducir automáticamente música, obtener información sobre cualquier tema, etc.

Recientemente la inteligencia artificial ha comenzado a incorporarse a la medicina para mejorar la atención al paciente y así acelerar los procesos y lograr una mayor precisión en los diagnósticos dando mejor atención médica. Esto puede fomentar comportamientos saludables para prevenir y reducir patologías en los pacientes.

Los algoritmos de aprendizaje automático han ayudado a esta evolución tecnológica por su gran eficiencia en aplicaciones de clasificación, predicción o detección, y por eso se han ido incorporando también en el ámbito sanitario. Con la ayuda de estos algoritmos, junto a la cantidad de información que poseen nuestras constantes vitales y las señales que podemos obtener de ellas, se pueden llegar a detectar muchas patologías sin necesidad de realizar pruebas intrusivas. Un ejemplo de ello, son las técnicas de diagnóstico para patologías en la voz. Estas técnicas suelen ser muy invasivas, ya que, en el caso de realizar una laringoscopia, el médico tiene que insertar un tubo a través de la nariz o de la boca, por lo que resulta molesto para el paciente. Así, gracias al aprendizaje automático estas técnicas pueden ser sustituidas y no sería necesario utilizarlas para detectar patologías en la voz.

1.2. Estado del arte

Actualmente, como hemos dicho antes, la tecnología llega a tener un papel fundamental en el ámbito de la salud, ya que puede llegar a mejorar la asistencia sanitaria de una manera más significativa y rápida. Puede desempeñar un mayor control en los comportamientos saludables de las personas y llegar a prevenir o reducir ciertas patologías.

La voz nos puede dar una gran cantidad de información sobre el hablante a través de cambios de tono vocal. Los oyentes pueden llegar a diferenciar en la voz aspectos como el género, la edad, la inteligencia, la educación, el nivel socioeconómico del hablante, etc. Así, la voz es única para cada persona y está compuesta por una serie de elementos que constituyen tanto la calidad como la eficacia de la propia voz.

Los sistemas automatizados para el diagnóstico y detección de patologías en la voz a partir de grabaciones de audio pueden ser un ejemplo útil a la hora de detectar, evaluar y realizar un

seguimiento de las enfermedades de los pacientes después de alguna terapia a la que estén sometidos.

Son varios los estudios que se han realizado sobre los sistemas automáticos para la detección de patologías en la voz. En el artículo realizado por el Departamento de Ingeniería y Ciencias de la Computación de la Universidad de Annamalai en la India [1], se centra en un sistema robusto, rápido y preciso para la detección automática del habla normal y patológica (identificando el tipo de esta última). Este sistema emplea medidas no invasivas, económicas y automáticas de las características del tracto vocal y la información de la excitación. Se crearon dos modelos para clasificar una señal de voz determinada, para luego, en el caso de que esa voz sea patológica, crear otros 10 modelos para clasificar diez tipos de patologías diferentes. Así, se podrá saber exactamente que patología es de las 10 definidas.

Los experimentos que se llevan a cabo en este artículo estudian el rendimiento del sistema para las características acústicas de los coeficientes cepstrales de predicción lineal (LPCC) y los coeficientes cepstrales de frecuencia Mel (MFCC) utilizando modelos de Markov y mezclas gaussianas.

Se llegó a la conclusión de que al clasificar el modelo de Markov con MFCC y con coeficientes delta y de aceleración presentaba una eficiencia de 94.44%, pero con el modelo de mezcla gaussiana aumentaba a un 95.74% de eficiencia. Así, el sistema propuesto en este artículo puede llegar a ser utilizado como una herramienta de gran importancia para investigadores y patólogos del habla para poder detectar si una señal de voz presenta ciertas patologías.

En otro artículo [2], realizado por el Instituto de Tecnología NMAN y el Instituto Nitte de Habla y Audición de la India, se han analizado trabajos de investigación relacionados con los sistemas de detección automática de trastornos de la voz (AVDD). El objetivo de este artículo es darnos una idea del tipo de investigaciones que se han realizado estos años atrás, a la vez que nos ayuda a destacar los puntos en los que se necesita mayor capacidad de experimentación y análisis. Esto ayudará a tener un sistema AVDD ideal, que puede ser preciso, eficiente y adecuado para poder detectar y diagnosticar lo antes posible diversos trastornos de la voz sin apenas esfuerzo.

Hay algunos artículos, como el de University College en Londres [3], que comparan el rendimiento de sistemas publicados para la detección automática de trastornos de la voz con una de las bases de datos que utilizaremos en este proyecto, la base de datos de voz de Saarbrücken (SVD). En el estudio, replican tres proyectos anteriores utilizando un protocolo de prueba común. Esos tres son: uno que utiliza un conjunto de características (OpenSMILE) y un clasificador (Support Vector Machine, SVM); otro con una entrada espectrográfica en una red neuronal convolucional (CNN); y, el último, utiliza una CNN entrenada con una arquitectura probada para el reconocimiento de imágenes (RESNET). Estos tres estudios tuvieron un porcentaje de acierto de 83%, 77% y 94%, respectivamente.

Se llegó a la conclusión de que el método OpenSMILE con SVM, tenía mejor rendimiento en patologías orgánicas (patologías físicas de la laringe) que haciendo uso tanto de las orgánicas como de las no orgánicas (patologías por uso ineficaz del mecanismo vocal). Aparte, el rendimiento con todas las vocales de un hablante es mejor que cuando solo se utiliza una sola vocal para cada hablante. Por eso, al tener más información con todas las vocales, se obtuvo un 85% de precisión para el subconjunto de patología orgánica. Por otro lado, los dos métodos de CNN obtuvieron un rendimiento similar entre sí y entre los métodos SVM; aunque este último

sigue siendo mejor, ya que los métodos de CNN se pueden ver más afectados por la reducción en el número de muestras de entrenamiento.

En conclusión, el trabajo planteado en este artículo es una forma de utilizar la base de datos SVD para entrenar y evaluar métodos automatizados para la detección de patologías en la voz, logrando que tres perspectivas diferentes tengan un rendimiento parecido utilizando las mismas patologías.

Otro artículo que hay que destacar, es el que nos describe el diseño y la construcción de la base de datos VOICED (VOice ICar fEDerico II), la cual también utilizaremos en este trabajo [4]. Está realizado por el "Instituto de Computación de Alto Rendimiento y Redes del Consejo Nacional de Investigación de Italia (ICAR-CNR)" y el Hospital Universitario de Nápoles "Federico II". Esta base de datos consta de 208 voces sanas y patológicas recogidas durante un estudio clínico siguiendo las pautas del protocolo médico SIFEL (Società Italiana di Foniatria e Logopedia) y la Declaración SPIRIT (Standard Protocol Items: Recommendations for Interventional Trials) de 2013. Se llega la conclusión de que, para producir una base de datos de calidad, hay que tener en cuenta la cantidad, la calidad y la disponibilidad de los datos. Más tarde, se hablará en más profundidad de esta base de datos, en el apartado Bases de Datos.

Y, para terminar este apartado del trabajo, comentar un artículo del Instituto Internacional de Tecnología de la Información de Hyderabad en la India [5]. Este artículo propone otro enfoque para la detección y evaluación automática de las patologías de la voz desde una perspectiva clínica. Se usa la base de datos de Saarbrücken y cuatro clasificadores binarios que han sido entrenados utilizando SVM con características de fuente de excitación, características del sistema de tracto vocal (MFCC) y características de OpenSMILE. Los estándares de OpenSMILE (ComParE y eGEMAPS) presentan un mejor rendimiento en términos de precisión de clasificación del 82.8% y 76%, respectivamente. Aunque, si las características de la fuente de excitación se combinan con las características de la línea de base mejoraron el rendimiento. Por ejemplo, según el artículo, si combinamos las características glotales y las características de ComParE de OpenSMILE, presentan un 85.2%, el cual es mayor que los dos anteriores.

Se realizan cuatro experimentos en este artículo: clasificación entre sanos y patológicos; clasificación de trastornos de la voz orgánicos y no orgánicos; clasificación de los trastornos de la voz estructurales y neurogénicos; clasificación de los trastornos de la voz funcionales y psicógenos. En este caso, el experimento que nos interesa para este proyecto es el primero, ya que se trata de una experimentación similar a la que realizamos en este TFG.

Como conclusión para este apartado, existe gran cantidad de artículos sobre el diseño e implementación de sistemas de detección de patologías en la voz utilizando aprendizaje automático. Esto sucede porque es muy probable que poco a poco el nivel de uso de estos sistemas aumente a medida que la tecnología siga evolucionando, ya que utilizan técnicas no invasivas y pueden llegar a tener un gran papel para poder investigar ciertas patologías y hacer el trabajo más fácil al ámbito sanitario.

1.3. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el de diseñar e implementar un sistema que sea capaz de detectar la presencia de patologías en la voz utilizando técnicas avanzadas de aprendizaje automático. Para llegar a este objetivo final, se han ido marcando diferentes objetivos intermedios:

- Documentación y búsqueda de información sobre el tema a tratar.
- Diseño del dataset para entrenamiento y test.
- Programación y experimentación con sistemas base de tres tecnologías de aprendizaje automático (GMM, SVM, DNN).
- Programación y experimentación con modelos basados en supervectores y embeddings.
- Evaluación de resultados

1.4. Herramientas

Para poder desarrollar el trabajo completo se ha utilizado, principalmente, Matlab, el cual es una plataforma de programación y cálculo numérico que se usa a lo largo del Grado de Ingeniería de Tecnologías y Servicios de Telecomunicación para analizar datos, desarrollar algoritmos y crear modelos.

Además, también se ha hecho uso de Python con la ayuda del programa PyCharm Community para obtener los parámetros de OpenSMILE, los cuales hablaremos más adelante.

1.5. Organización de la memoria

La memoria del trabajo está dividida en:

- Capítulo 1: Introducción. En este primer capítulo se introduce el tema del trabajo, así como su motivación, objetivos, herramientas utilizadas y ejemplos de otros artículos que han desarrollado el mismo tipo de tema.
- Capítulo 2: Bases de datos. Se realiza un análisis detallado de las dos bases de datos utilizadas en este trabajo.
- Capítulo 3: Sistemas de detección de patologías. Se divide en dos secciones. Una sección donde veremos los parámetros utilizados dando a conocer sus características y la forma de obtenerlos. Y otra donde se desarrollarán los tres clasificadores que se utilizan.
- Capítulo 4: Resultados. Analizaremos los resultados que hemos obtenidos de todas las pruebas realizadas.
- Capítulo 5: Conclusiones. Se detallarán las conclusiones de todo el trabajo realizado.
- Capítulo 6: Referencias. Se muestra toda la bibliografía utilizada en el proyecto.

- Capitulo 7: Anexos. Se muestra el código de varios programas utilizados en este trabajo para llegar a obtener los resultados deseados.

2. BASES DE DATOS

Los sistemas de detección de patologías en la voz cada vez son más utilizados en el ámbito sanitario para poder detectar, evaluar y realizar un seguimiento al paciente. Para que esto sea posible, se necesitan bases de datos de alta calidad.

Para este proyecto han sido utilizadas dos tipos de bases de datos ampliamente utilizadas en los trabajos de investigación: VOICED y SVD.

2.1. Base de datos VOICED:

Esta base de datos ha sido realizada por el "Instituto de Computación de Alto Rendimiento y Redes del Consejo Nacional de Investigación de Italia (ICAR-CNR)" y el Hospital de la Universidad de Nápoles "Federico II" [4].

Consta de un total de 208 voces que están divididas en 73 voces masculinas y 135 voces femeninas. Predominan las voces patológicas frente a las voces saludables. En total hay 150 patológicas (52 masculinas y 98 femeninas) y 58 voces saludables (21 masculinas y 37 femeninas). Esto es debido a que las mujeres tienen más iniciativa a participar en este tipo de estudios y, también, presentan mayor incidencia de trastornos en la voz.

La edad promedio de esta base de datos está en torno a los 40 años tanto en mujeres como en hombres, siendo entre los 40 y 60 años la categoría de edad con más trastornos en la voz.

Las voces patológicas son de tres tipos: disfonía hipercinética, disfonía hipocinética y laringitis por reflujo [4].

- Disfonía hipercinética: este trastorno está caracterizado por una hipercontracción muscular del aparato neumo-fónico, una patología común en trabajos que requieren un esfuerzo extra en la voz. Puede producirse al forzar la voz o elevarla más de lo normal, haciendo que la fonación sea más agotadora, lo que da lugar a una alteración de la dinámica respiratoria. Las enfermedades que puede causar este trastorno son: nódulos de las cuerdas vocales, el edema de Reinke, pólipos, cuerdas vocales rígidas, etc.
- Disfonía hipocinética: se caracteriza por una reducción de la aducción de las cuerdas vocales durante el ciclo respiratorio, más concretamente en la fase inspiratoria, produciendo una obstrucción del flujo de aire a la altura de la laringe. Al no cerrarse las cuerdas vocales, se produce una voz débil y sin aliento. Con este trastorno, la voz mejora si se aumenta su intensidad, pero puede ocasionar un problema vocal. Este tipo de disfonía puede causar déficit de aducción, presbifonía, insuficiencia glótica, parálisis en las cuerdas vocales, etc.
- Laringitis por reflujo: es una inflamación de la laringe provocada por la acumulación de ácido del estómago en el esófago. Uno de los síntomas más comunes es una ronquera crónica, pero hay otros síntomas que pueden ser faringitis, asma, halitosis, etc.

Hay que destacar, que esta base de datos es la primera que contiene información sobre hábitos de vida de los pacientes (como su consumo de alcohol, si es fumador, su hidratación,

hábitos alimenticios, etc.), el carácter de estos, antecedentes o enfermedades, y los resultados de dos cuestionarios populares (Voice Handicap Index (VHI) y Reflux Symptom Index (RSI)) utilizados para autoevaluar la voz y las consecuencias psicosociales de los trastornos de la voz. Esto es interesante conocerlo ya que, por ejemplo, el consumo de alcohol puede provocar una disminución en el control y coordinación de la articulación del habla, fonación y respiración. También el café, chocolate, quesos suaves, bebidas carbonatadas, tomates y cítricos pueden provocar reflujo.

2.2. Base de datos de Saarbrücker (SVD):

Esta base de datos ha sido realizada por la Universidad de Saarland, más concretamente, por el Instituto de Fonética.

Contiene más de 2000 grabaciones de voz en formato ".wav" con una frecuencia de muestreo de 50 kHz y 16 bits. Está dividida en 687 voces de personas sanas (428 mujeres y 259 hombres) y 1356 voces patológicas (629 hombres y 727 mujeres). En esta base de datos tenemos un total de 71 trastornos de voz diferentes.

Están presentes tres tipos de grabaciones:

- Grabaciones con sonidos vocales sostenidos (a, u, i) en tono bajo, normal y alto.
- Grabaciones con sonidos vocales sostenidos (a, u, i) en tono ascendente y/o descendente.
- Grabaciones de una frase en alemán: "Guten Morgen, wie geht es Ihnen?" ("Buenos días, ¿cómo estás?").

Esta base de datos es de uso público (http://www.stimmdatenbank.coli.uni-saarland.de/index.php4#target). En la página web anterior, podemos escoger las voces que queremos tener, ya que podemos elegir entre el sexo, tipo de disfonía, rango de edad de las personas, etc.

3. SISTEMAS DE DETECCIÓN DE PATOLOGÍAS

En este apartado se explicarán tanto los parámetros que se han extraído en este trabajo como los clasificadores utilizados. En la Figura 1, podemos ver un diagrama de bloques que resume este trabajo.

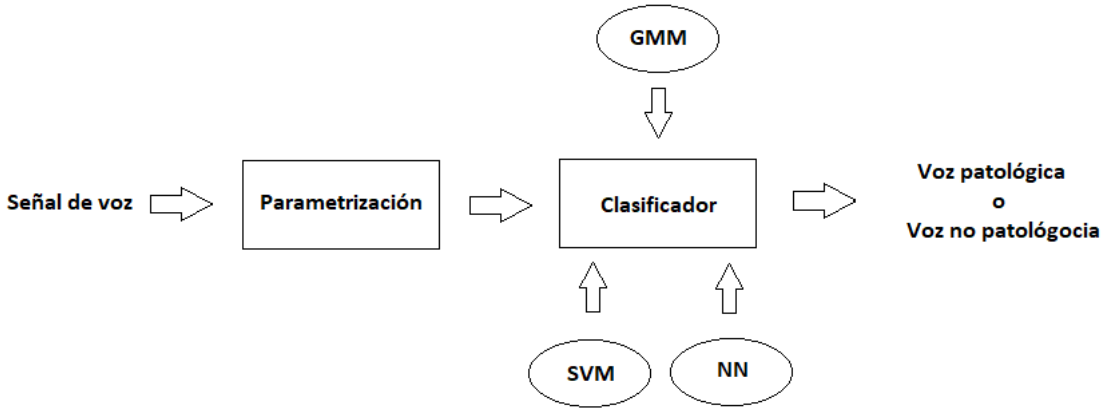

Figura 1. Diagrama de bloques

3.1. Parámetros

3.1.1. MFCC: Mel Frequency Cepstrum Coefficients

Los parámetros MFCC son coeficientes que provienen del dominio cepstral y se caracterizan por tener gran cantidad de información de la voz. Formalmente, el cepstrum es la transformada inversa del logaritmo de la transformada de Fourier de una secuencia. Como propiedad más característica es que transforma las convoluciones en sumas. Como la señal de voz es el resultado de una convolución de una señal de excitación y un filtro variante en el tiempo (tracto vocal), en el dominio cepstral ambas señales quedan separadas. Por otro lado, la percepción de frecuencia en el ser humano sigue una escala no lineal denominada Mel. La escala es más o menos lineal a baja frecuencia y logarítmica en alta frecuencia. Así, los parámetros Mel-Cepstrum nos dan una representación compacta de la señal de voz.

Estos coeficientes son muy utilizados en el análisis de la señal de voz para aplicaciones de reconocimiento del habla o del hablante, entre otras.

El procedimiento para calcular los coeficientes Mel se muestra en el diagrama de bloques de la Figura 2. Consta de cinco procesos:

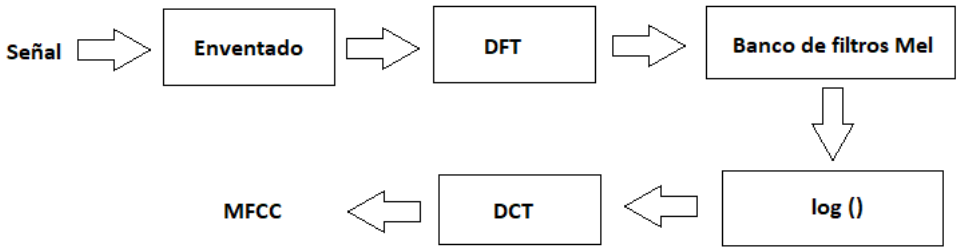

Figura 2. Diagrama de bloques para obtener los MFCC

1. Enventanado: se realiza un enventanado de la señal, ya sea con una ventana rectangular, una ventana de Hamming o una ventana de Hanning. En la Figura 3, podemos ver el diagrama de las tres ventanas, donde se ve que la ventana de Hanning tiene una ligera caída en el borde a diferencia de la de Hamming, que tiene una caída brusca. En este trabajo se ha utilizado una ventana de 30 ms de duración y un desplazamiento de 5 ms.

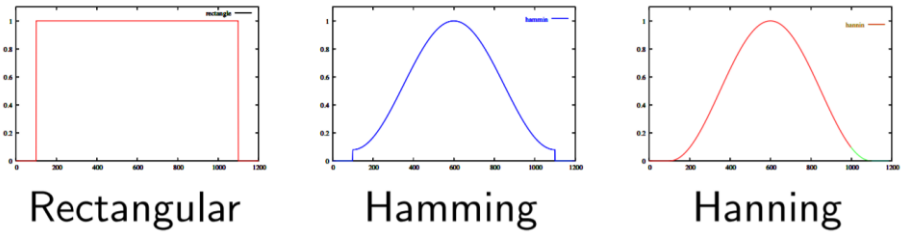

Figura 3. Representación de las ventanas

2. Transformada discreta de Fourier (DFT): a la señal resultante del enventanado se le aplica la DFT (ecuación 1) para extraer información en el dominio frecuencial.

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(-j\frac{2\pi}{N}kn\right) \quad [1]$$

3. Banco de filtros Mel: realizamos un filtrado en el dominio frecuencial con un banco de filtros triangular de área unidad (Figura 4) para poder llegar a imitar lo que percibe un ser humano. Es decir, se intenta imitar cómo la membrana basilar del oído detecta la vibración de cada sonido. Así, el paso de banda triangular es más ancho en las frecuencias altas para reflejar esa propiedad de la percepción auditiva humana y hacerlo menos sensible en esas frecuencias. Como vemos en la Figura 5, y como bien hemos comentado antes, la escala Mel que se le aplica a estos filtros es lineal por debajo de 1000 Hz y logarítmica después (ecuación 2), ya que el volumen percibido cambia según la frecuencia en los humanos.

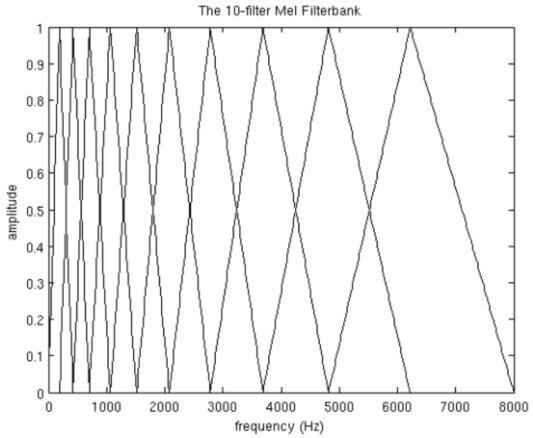

Figura 4. Banco de filtros triangular

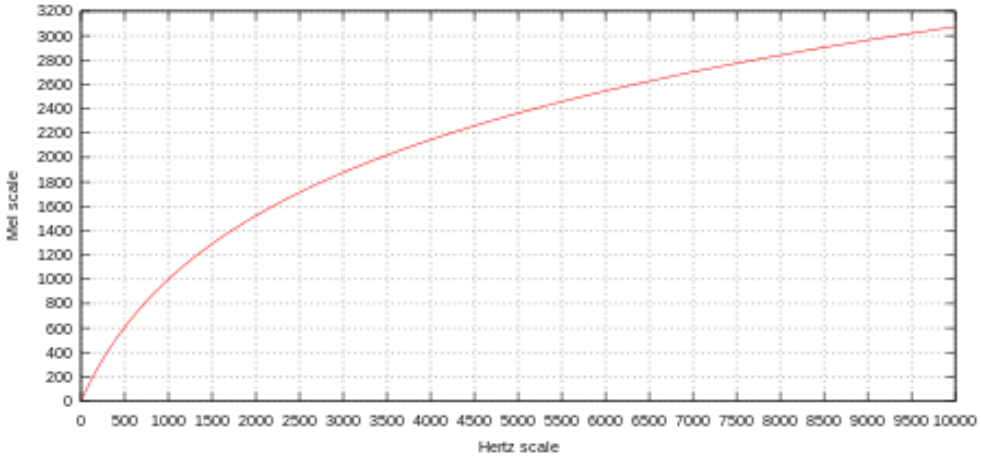

Figura 5. Gráfico de la escala de frecuencias Mel

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$ [2]

4. Logaritmo: se calcula el logaritmo de la energía correspondiente a la salida de los filtros. Esto se realiza porque los seres humanos son menos sensibles a los pequeños cambios a alta energía que a los pequeños cambios a bajo nivel de energía. También se llega a reducir las variantes acústicas que no son significativas para el reconocimiento de la voz.
5. Discrete Cosine Transform (DCT): se aplica la DCT para poder pasar al dominio cepstral para obtener los parámetros MFCC.

Una vez que tenemos los parámetros MFCC, solo necesitamos los primeros coeficientes porque es ahí donde está la mayor cantidad de información.

3.1.2. OpenSMILE

OpenSMILE (open-source Speech and Music Interpretation by Large-space Extraction) es un conjunto de herramientas completo y de código abierto que nos ayuda a extraer características para aplicaciones de procesamiento de señales y aprendizaje automático. Su objetivo principal es obtener esas características de las señales de audio, pero también es posible utilizarlo para analizar señales de otras modalidades, como señales fisiológicas, señales visuales, etc. Está escrito únicamente en C++, posee una arquitectura rápida, eficiente y flexible, y se ejecuta en plataformas como Linux, Windows, MacOS, Android e IOS.

OpemSMILE tiene la capacidad de leer y escribir en varios formatos de datos que se usan habitualmente en el campo de las bases de datos y el aprendizaje automático. Estos formatos incluyen PCM WAVE para archivos de audio, CSV (formato de hoja de cálculo) y ARFF (Weka Data Mining) para archivos de datos basados en texto, archivos de parámetros HTK (Hidden-Markov Toolkit) y un formato de matriz flotante binaria simple para datos de características binarias.

En cuanto a los estándares que tiene OpenSMILE, se admiten tres: ComParE 2016, GeMAPS y eGeMAPS. Aunque en este trabajo solo serán utilizados el primero y último, los cuales vamos a desarrollar a continuación.

3.1.2.1. ComParE

El estándar ComParE está basado en un sistema de extracción automática de características que nos da ciertos parámetros de una señal de audio. Antes de esta versión que vamos a utilizar en este trabajo, ha habido varios estudios sobre las características del habla que han sido redactados por los mismos autores que el artículo [6]. En este mismo artículo se abordan tres nuevos problemas dentro del campo de la paralingüística computacional: la identificación del discurso engañoso, el grado de sinceridad percibida en los hablantes y el grado de natividad.

El ComParE es un conjunto de características del desafío de Paralingüistica Computacional Interspeech 2016 ([6]) que contiene 6373 características estáticas resultantes del cálculo de varias funciones sobre contornos de LLD (Low Level Descriptors).

3.1.2.2. eGeMAPS

El estándar eGeMAPS se basa en un sistema de extracción automática que extrae un conjunto de parámetros acústicos de una forma de onda de audio sin interacción o corrección manual. Solo se han incluido parámetros que se pueden extraer de forma fiable y sin supervisión en condiciones acústicas limpias.

Este estándar se basa en un estudio realizado y concebido en una reunión interdisciplinaria de un conjunto de científicos de la voz y el habla en Ginebra y desarrollado en Technische Universität München (TUM) [7]. Es un conjunto de parámetros acústicos elegidos con tres criterios: el potencial de un parámetro acústico para indexar cambios fisiológicos en la producción de voz durante los procesos afectivos; la frecuencia y el éxito con el que el parámetro se ha utilizado; y su significado teórico.

Este conjunto de parámetros estándar es minimalista, ya que está compuesto por 88 parámetros, a diferencia del estándar ComParE, que tiene más de 6000. Este conjunto de parámetros acústicos contiene un conjunto compacto de 18 LLD (Low Level Descriptors), ordenados por grupos de parámetros:

- Parámetros relacionados con la frecuencia: pitch, jitter, frecuencia de los formantes 1, 2 y 3, ancho de banda del formante 1, 2 y 3.
- Parámetros relacionados con la energía o amplitud: shimmer, intensidad de la señal, relación de armónicos a ruido (Harmonics to Noise Ratio, HNR).
- Parámetros espectrales: relación de la energía sumada de 50 a 1000 Hz y de 1 a 5 kHz, índice de Hammarberg, pendiente espectral en las bandas 0-500 Hz y 500-1500 Hz, energía relativa de los formantes 1, 2 y 3, relación de energía del primer y segundo armónico, relación de energía del primer formante a la energía del armónico más alto en el tercer rango de formantes, flujo espectral, MFCC del 1 al 4.

3.1.3. Glotal

Los parámetros glotales [5] son obtenidos a partir de la forma de onda del flujo glotal que se estima a partir del análisis de fase cuasi-cerrada (QCP). Estos parámetros incluyen características del dominio temporal llamados cocientes abiertos (OQ1, OQ2), cocientes cercanos (CQ, CQa), cocientes de velocidad (SQ1, SQ2) y cocientes de amplitud (AQ, NAQ, QOQ). También presenta parámetros del dominio frecuencial como la diferencia de amplitud entre el primero y segundo armónico glotal (H1-H2), el parámetro espectral parabólico (PSP) y el factor de riqueza armónica (HRF). Estos parámetros frecuenciales son extraídos a partir del espectro de la forma de onda glotal. Así, obtenemos el vector de características glotales de 192 dimensiones para cada forma de onda.

3.1.4. Entonación

Los parámetros de entonación [5] es un vector de características que incluye: 5 estadísticas de la frecuencia fundamental (F0), 22 parámetros de la energía de excitación (EoE), 22 parámetros de fluctuación de F0, 22 parámetros de brillo EoE, 4 parámetros de relación de

armónicos a ruido y una media entropía de perturbación de tono. Así, obtenemos un vector de 76 dimensiones que se denomina vector de características de entonación.

3.2. Clasificadores

La clasificación de datos es una tarea típica en el aprendizaje automático, ya que tendremos que decidir si un conjunto de datos pertenece a una clase o a otra.

En este apartado comentare los tres algoritmos de clasificación utilizados en este trabajo: Gaussian Mixture Models (GMM), Support Vector Machine (SVM) y redes neuronales.

3.2.1. GMM

Un modelo de mezcla gaussiana (GMM) es una función de densidad de probabilidad paramétrica representada como una suma ponderada de las densidades de las componentes gaussianas. Los GMM se suelen utilizar como modelos paramétricos de las distribuciones de probabilidad de ciertas medidas o características continuas en sistemas biométricos, como las características espectrales relacionadas con el tracto vocal en un sistema de reconocimiento de voz.

Este tipo de modelos probabilísticos supone que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones gaussianas con parámetros desconocidos. Se puede pensar en los modelos de mezcla como agrupaciones de k-medias generalizadas para incorporar información sobre la estructura de covarianza de los datos, así como los centros de los gaussianos latentes.

El algoritmo que usa GMM pertenece a los algoritmos de aprendizaje no supervisado, donde se basan en la probabilidad para poder asignar si las componentes pertenecen a un grupo o a otro.

En nuestro caso, como veremos más adelante, utilizaremos el GMM para formar dos agrupaciones y comprobar si las divisiones que haga el algoritmo coinciden con las etiquetas de los datos de entrada.

3.2.2. SVM: Support Vector Machine

Las SVM (Support Vector Machine) son modelos de aprendizaje supervisado con algoritmos de aprendizaje que analizan datos para clasificación y análisis. Al basarse en marcos de

aprendizaje estadísticos, es uno de los métodos más robusto para predecir. Si tenemos una serie de datos de entrenamiento con dos clases diferentes, el algoritmo de entrenamiento de SVM formará un modelo que asignará nuevos datos a una clase u otra, convirtiéndolo en un clasificador lineal binario no probabilístico. Esto hace que se aumente la diferencia entre las dos clases y se cree una división entre ellas. Luego, los datos que ha incluido, los localiza en ese espacio y se hace una predicción para saber a qué clase pertenecen según estén situados a un lado o al otro de la división que se crea anteriormente.

Con SVM se pueden realizar de manera sencilla una clasificación no lineal mapeando, de forma implícita, sus entradas en espacios de características de alta dimensión.

En el caso de que los datos no estén etiquetados, es decir, que no sepamos a que clase pertenecen, el aprendizaje supervisado, que hemos hablado anteriormente, no sería posible. Así, pasaríamos a un aprendizaje no supervisado, intentando localizar la agrupación natural de los datos en grupos, para después localizar los nuevos datos y decidir a qué grupo pertenecen.

De forma más formal, una SVM construye un hiperplano o un conjunto de hiperplanos en un espacio de dimensión alta o infinita que puede usarse para clasificación, regresión u otras tareas como la detección de valores atípicos. Se llega a lograr una separación por el hiperplano que tiene la mayor distancia al punto de datos de entrenamiento más cercano de cualquier clase (margen funcional), ya que, si el margen es mayor, menor será el error de generalización del clasificador.

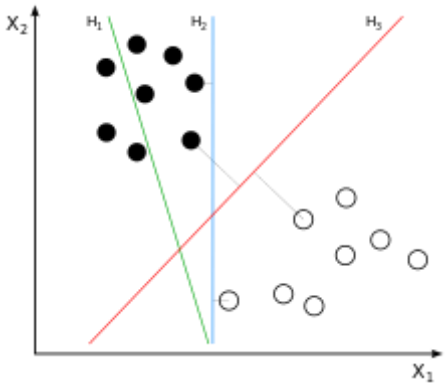

Figura 6. Ejemplos de tipos de división realizadas por el SVM, donde la línea H3 sería la indicada para realizar la división de los datos

3.2.3. Redes neuronales

Una red neuronal es un conjunto de neuronas interconectadas con otras neuronas que forman una red lo más parecido posible a una representación del cerebro. Este tipo de redes pueden aprender de los datos, de manera que es posible entrenarla para reconocer patrones, clasificar datos y acertar en eventos futuros.

El funcionamiento de una red neuronal es simple, ya que se ingresan unas determinadas variables como entrada y, después de varios cálculos, obtenemos una salida. Su comportamiento se define por la forma y la importancia de las conexiones de las neuronas.

Las redes neuronales son adecuadas a la hora de reconocer patrones para identificar y clasificar objetos o señales en sistemas de voz, visión y control. En el aprendizaje automático, este tipo de red tienen muchas capas, a veces cientos. Se combinan diversas capas de procesamiento y utiliza elementos simples que operan en paralelo, inspirándose en los sistemas nerviosos biológicos. Primero presenta una capa, la de entrada, seguida de una o varias capas ocultas, para acabar en la capa de salida. Estas capas están interconectadas por nodos o neuronas, por lo que cada capa utiliza la salida de la capa anterior como entrada. En la Figura 7 se puede ver un ejemplo de una arquitectura típica de una red neuronal.

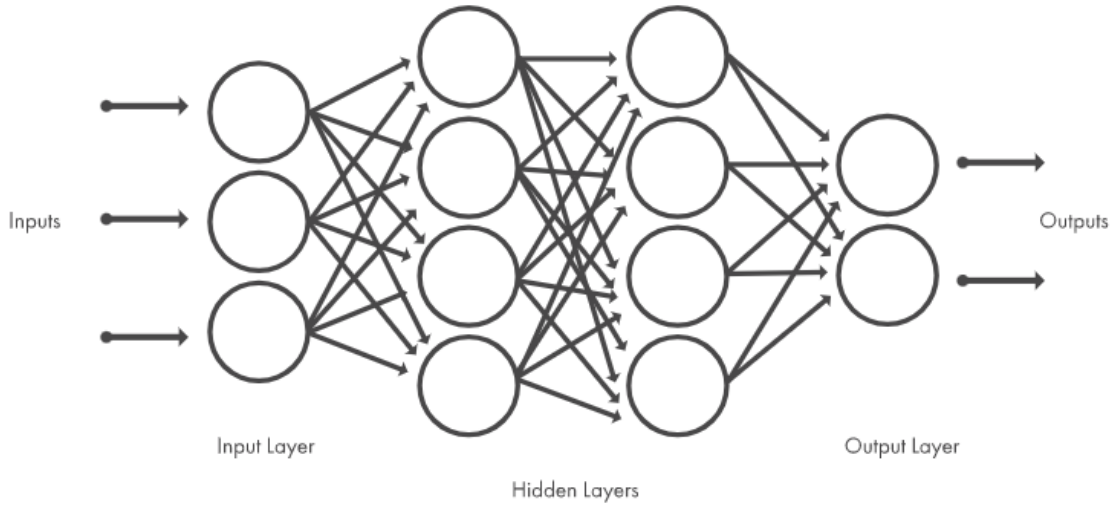

Figura 7. Arquitectura de una red neuronal

Hay diversas técnicas de aprendizaje automático para diseñar aplicaciones de redes neuronales como el aprendizaje supervisado y no supervisado, la clasificación, la regresión, el reconocimiento de patrones y el clustering.

- Las redes neuronales supervisadas se entrenan para obtener las salidas que el usuario quiere como respuesta a la entrada, por lo que son idóneas para modelar y controlar sistemas dinámicos, clasificar datos con ruido y predecir eventos futuros.
- La clasificación es un tipo de machine learning supervisado en el que un algoritmo puede llegar a clasificar nuevos datos a partir de ejemplos con datos etiquetados.
- Los modelos de regresión nos dan la relación entre la salida y la entrada.
- El reconocimiento de patrones funciona mediante la clasificación de datos de entrada en objetos o clases mediante la clasificación supervisada o no supervisada.
- Las redes neuronales no supervisadas se entrenan permitiendo que la red neuronal se autoajuste a las nuevas entradas. Se suelen utilizar en el caso de que los datos de entrada no estén etiquetados.
- El clustering es un enfoque de aprendizaje no supervisado en el que se pueden emplear redes neuronales para el análisis de datos exploratorio a fin de localizar patrones ocultos o agrupaciones de datos. Se suelen agrupar los datos por similitud entre ellos.

Hay que destacar que las operaciones que hace cada neurona de la red son muy simples. Como podemos ver en la Figura 8, el valor de cada neurona de la columna anterior se multiplica por unos pesos, los cuales determinan la conexión entre las dos neuronas, que se modificarán durante el proceso de aprendizaje, para después sumar todas esas multiplicaciones. Tras esta suma, la neurona aplica una función llamada función de activación. Esta función nos ayuda a convertir el valor calculado a un número entre 0 y 1. Hay cuatro tipos principales de funciones de activación: funciones de umbral, funciones sigmoideas (Figura 9), funciones rectificadoras o ReLUs y funciones de la tangente hiperbólica.

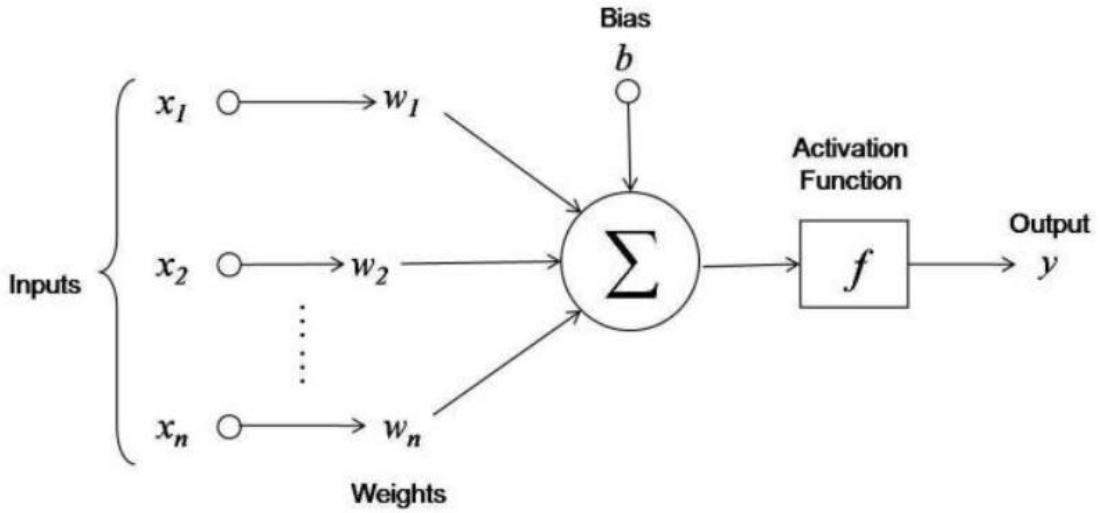

Figura 8. Operaciones realizadas por una neurona

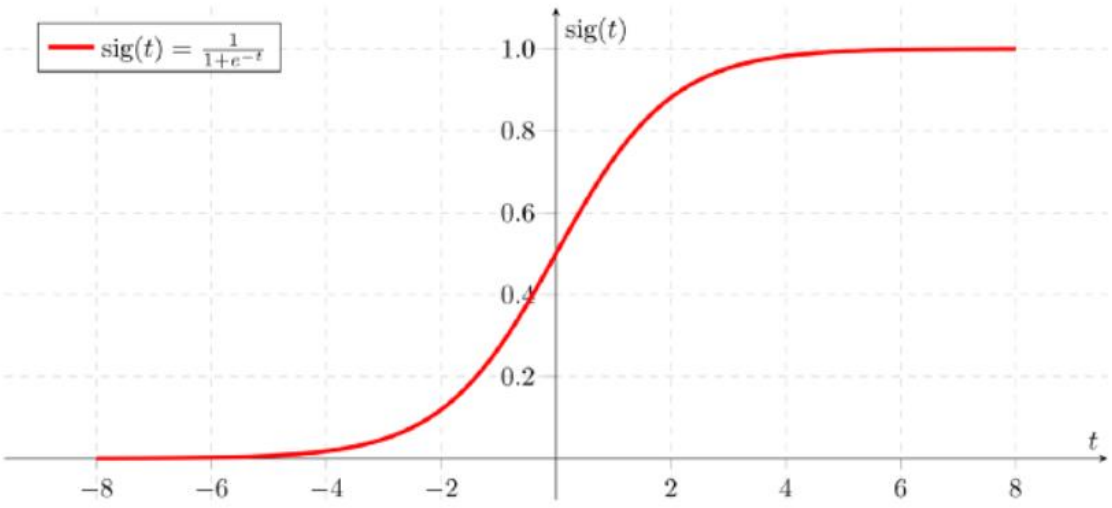

Figura 9. Ejemplo de función de activación sigmoidea

4. RESULTADOS

En este aparatado vamos a ver los resultados que se han obtenido de los diferentes clasificadores realizados, así como el proceso para obtener esos resultados.

4.1. GMM

El primer clasificador que vamos a usar son los modelos de mezcla gaussiana (GMM) de los cuales obtendremos tanto la curva de rendimiento (curva ROC), para saber la eficacia de este clasificador, como el área bajo la curva (AUC). En el caso de que el AUC sea igual a 1, significa que el clasificador es perfecto.

El procedimiento de obtención de estas curvas ROC es el mismo en todas las pruebas realizadas y sigue los siguientes pasos:

1. Extracción de parámetros MFCC: realizamos una división de los datos de entrada separándolos entre voz con patologías o sin patologías. Una vez que se hayan creado esa división, obtenemos los coeficientes MFCC, como lo hemos explicado en el apartado 3.1.1., de cada clase (patológicas o no patológicas) y creamos dos vectores con sus medias y su desviación estándar.
2. Entrenamiento del modelo GMM: para obtener los modelos de mezcla gaussiana hacemos uso de la función de Matlab 'fitgmdist'. Para esta función serán necesarios los vectores que hemos obtenido anteriormente y elegir el número de gaussianas que queremos. Así, tendremos un modelo de voz con patología y otro sin patología.
3. Extracción de estadísticas con los vectores de entrenamiento: extraemos los ratios de verosimilitud para los dos modelos para saber las posibilidades de que cada voz sea patológica o no con la ayuda de los modelos. Para ello obtenemos la función de densidad de probabilidad (PDF) en escala logarítmica de cada fichero de voz con y sin voz patológica con los dos modelos. De esta manera, y como vemos en la Ecuación 3, obtenemos los ratios de verosimilitud.

$$Ratio\ de\ verosimilitud = \log\left(PDF_{modelo_{patológico}}\right) - \log\left(PDF_{modelo_{no-patológico}}\right) \quad [3]$$

4. Obtención curva ROC: hacemos uso de la función de Matlab 'perfcurve', donde serán necesarios: un vector de etiquetas (1 es patológico y si es 0 es sano) y un vector donde estén los ratios de verosimilitud correspondientes a las etiquetas. Obtenemos tanto la curva ROC como el área bajo la curva (AUC) y el punto de operación óptimo.

A continuación, vamos a ver diferentes casos en los que ponemos a prueba al clasificador. El primero de ellos es hacer uso de las dos bases de datos completas. Es decir, hacer una división entre patológicas y no patológicas de las dos bases, por separado, y seguir el procedimiento descrito arriba. En las Tablas 1 y 2, podemos ver los valores de AUC y los puntos de operación óptimos (OPTROC) obtenidos con el clasificador variando el número de gaussianas para crear los modelos para las dos bases de datos. Se puede observar que el valor de AUC aumenta a

medida que el número de gaussianas es mayor. Esto se da porque a mayor número de gaussianas, más facilidad a la hora de separar entre las dos clases, aunque hasta cierto límite, ya que en el caso de la base de datos VOICED, se consigue un clasificador perfecto o casi perfecto con 32 gaussianas. Por lo tanto, el número de gaussianas es importante controlarlo ya que puede mejorar el rendimiento del clasificador.

K	1	2	4	8	16	32	64	128
AUC	0.7083	0.71954	0.7397	0.7244	0.7658	0.8203	0.8998	0.9562
OPTROC	0.763, 0.92	0.644, 0.874	0.628, 0.886	0.6177, 0.857	0.5799, 0.883	0.484, 0.883	0.339, 0.927	0.170, 0.919

Tabla 1. AUC y OPTROC con la base de datos SVD

K	1	2	4	8	16	32
AUC	0.6446	0.684	0.7595	0.8745	0.96224	1
OPTROC	0.877, 0.960	0.947, 1	0.789, 0.993	0.439, 0.927	0.193, 0.967	0, 1

Tabla 2. AUC y OPTROC con la base de datos VOICED

El segundo caso, es realizar particiones en las dos bases de datos. Para este trabajo se ha optado por separar en diez grupos la base de datos VOICED y en cinco la SVD, de forma equilibrada. Cada partición, a su vez, también estará dividida entre patológicas y no patológicas para seguir el mismo proceso de obtención de la curva ROC descrito anteriormente, pero con cierta modificación. En este caso, se obtiene un vector de medias y desviación de los coeficientes de MFCC, pero no de todas las particiones ya que nos dejamos una partición aparte que será la partición de test. Esta partición de test tendrá su propio vector. Así, creamos los modelos de entrenamiento con el primer vector (el vector que no tiene la partición de test), para después obtener los ratios de verosimilitud con estos modelos y el vector obtenido con la partición de test. Y, de la misma forma que antes, obtenemos tanto la curva ROC como el AUC.

En las Tablas 3 y 4, vemos los valores del AUC, donde en las filas varía el número de gaussianas (K) para obtener los modelos y en las columnas varía la partición usada para test. He de destacar que cada valor de AUC es una media de diez iteraciones para obtener un resultado más global. A diferencia del caso anterior, en el cual, si aumentabas el número de gaussianas, el clasificador mejoraba su rendimiento, en este caso ese rendimiento permanece constante. Esto es debido a que antes utilizábamos toda la base de datos tanto para entrenar como para test, por lo que era más fácil acertar al aumentar el número de gaussianas. Aquí la partición de test no está presente en el entrenamiento, por lo que el clasificador lo tiene más difícil, pero, aun así, en la base de datos de SVD, presenta una alto porcentaje de rendimiento. En cambio, con la base de datos VOICED, al ser más pequeña y no tener tantos parámetros para entrenar el valor del AUC es muy variante dependiendo de la partición de test y puede llegar a fallar más que con la otra base de datos.

K \ Test	1	2	3	4	5
1	0.7001	0.6946	0.6659	0.7027	0.7159
2	0.7015	0.7009	0.6983	0.6986	0.7306
4	0.7093	0.6920	0.6836	0.7079	0.7138
8	0.7201	0.6673	0.6615	0.6810	0.7137
16	0.7163	0.7053	0.6656	0.6810	0.7032
32	0.7057	0.6944	0.6411	0.6811	0.6888
64	0.6433	0.6850	0.6799	0.6781	0.6841
128	0.6668	0.6801	0.6871	0.6514	0.6692

Tabla 3. AUC con la base de datos SVD con particiones

K\Test	1	2	3	4	5	6	7	8	9	10
1	0.4889	0.5667	0.1889	0.4778	0.4750	0.5125	0.4444	0.5111	0.6400	0.5889
2	0.6078	0.4844	0.4256	0.5078	0.4825	0.4513	0.4322	0.4567	0.3787	0.4011
4	0.5211	0.4367	0.4356	0.5933	0.4725	0.4637	0.4467	0.4556	0.3667	0.4956
8	0.4189	0.5633	0.3211	0.6211	0.4850	0.5037	0.4500	0.5311	0.4587	0.5744
16	0.4511	0.6689	0.4478	0.4567	0.5488	0.5263	0.5033	0.5278	0.4987	0.5878
32	0.5278	0.6367	0.3944	0.4522	0.6137	0.6100	0.5711	0.4733	0.5600	0.5267

Tabla 4. AUC con la base de datos VOICED con particiones

Tras ver estos dos casos que hemos realizado con este clasificador, se puede llegar a la conclusión de que con una base de datos suficientemente grande y con un número de gaussianas apropiado podemos llegar a construir un clasificador decente con modelos de mezcla gaussiana.

4.2. SVM

El segundo clasificador que se va a usar es el Support Vector Machine (SVM). Para este clasificador se utilizan varias funciones de Matlab para obtener las tasas de error. El procedimiento para la obtención de esas tasas de error es el siguiente:

1. Obtención de etiquetas: al igual que en el anterior clasificador, si la entrada es patológica tendremos un uno y si es no patológica tendremos un cero.
2. Entrenamiento del clasificador: se entrenan los datos de entrada y sus respectivas etiquetas con ayuda de la función de Matlab 'fitcsvm', la cual devuelve un modelo entrenado.
3. Validación cruzada: se realiza una validación cruzada de 10 veces (10-fold), para el modelo obtenido antes, con la ayuda de la función de Matlab 'crossval', obteniendo un modelo de clasificación con validación cruzada.
4. Pérdida de clasificación: se obtiene la pérdida de clasificación para el modelo anterior.
5. Predicción: con los datos de entrada, el modelo entrenado obtenido en el punto 2 y con la función de Matlab 'predict', se obtiene la predicción de este clasificador al obtener un vector similar al de las etiquetas.
6. Tasa de error sobre training: utilizando la ecuación 4, podemos obtener las tasas de error sobre training.

$$Tasa\ de\ error\ sobre\ training = \frac{\Sigma\,|(etiquetas_{reales} - etiquetas_{clasificador})|}{length(etiquetas_{reales})} \quad [4]$$

Se han realizado tres pruebas para este clasificador con las dos bases de datos y con diferentes datos de entrada. Tenemos tres tipos de datos de entrada: un vector con la media y desviación estándar de los coeficientes MFCC (vector 1), un vector obtenido mediante adaptación MAP para crear supervectores con la ayuda de un modelo de mezcla gaussiana (vector 2) y un vector que realiza una reducción de parámetros del vector anterior quedándose con los valores más representativos (vector 3).

Primero vamos a utilizar la base de datos VOICED con los diferentes vectores. Para obtener el vector 1, se tiene que emplear el mismo método empleado con el clasificador GMM. Una vez que se cree ese vector, se utiliza como entrada al SVM siguiendo el proceso descrito anteriormente en este apartado. En este caso, se obtiene una tasa de error sobre el training de 0.2740, por lo que tiene un porcentaje de acierto del 72.60%. Aunque se repita el procedimiento varias veces, el resultado es constante porque no se modifican los valores de entrada en ningún momento, pero el clasificador tiene un porcentaje aceptable para funcionar.

Para obtener el vector 2, es necesario crear un modelo de mezcla gaussiana con la ayuda de la función de Matlab 'fitgmdist' como con el clasificador GMM. Con el modelo obtenido se llega a obtener un supervector utilizando adaptación MAP, el cual es nuestro vector 2. En la Tabla 5, se pueden ver los valores conseguidos modificando el número de gaussianas a utilizar para crear el modelo de mezcla gaussianas. Se puede observar, al igual que el clasificador GMM, cuanto mayor sea el número de gaussianas, más facilidad tiene el clasificador para acertar el resultado.

Y, para extraer el vector 3 a partir del vector 2, utilizamos un umbral para poder extraer los diferentes parámetros más relevantes de este último y, así, reducir las dimensiones del vector y tener mejor eficiencia. Para ello se necesita marcar un umbral adecuado para reducir esas dimensiones, pero sin perder la efectividad del clasificador. En la Tabla 5, se pueden ver las tasas de error de training con este último vector. Vemos que hay un punto en el que el clasificador tiende a ser peor con el vector de dimensiones reducidas que con el vector 2. Una razón por la que ocurre esto es por el umbral, ya que hay que definirlo dependiendo de los niveles obtenidos anteriormente y en cada caso puede cambiar ese punto óptimo. Pero una cosa a favor del uso

de estos dos vectores es que presenta mayor porcentaje de acierto que solo haciendo uso de las medias y desviaciones de los coeficientes MFCC.

Número de gaussianas	1	2	4	8	16	32
Vector 2	0.274038	0.211538	0.149038	0.057692	0.014423	0.014423
Vector 3	0.274038	0.211538	0.149038	0.177885	0.096154	0.211538

Tabla 5. Tasa de error sobre training con el vector 2 y 3

Ahora, si se hace uso de la base de datos SVD con los tres vectores anteriores y los mismos procedimientos, se sigue la misma tendencia de antes. Con el primer vector tenemos una tasa de error sobre training de 0.2918, lo cual es un valor aceptable y similar al conseguido con la base de datos VOICED. En la Tabla 6, podemos ver las tasas de error sobre training con los vectores 2 y 3 como entrada. Al igual que antes, la tasa de error sigue una tendencia descendente a medida que aumentamos el número de gaussianas.

Número de gaussianas	1	2	4	8	16	32	64	128
Vector 2	0.299657	0.292300	0.292791	0.281511	0.242276	0.101079	0.175576	0.133399
Vector 3	0.299657	0.292300	0.292791	0.281511	0.242276	0.215302	0.187347	0.162334

Tabla 6. Tasa de error sobre training con el vector 2 y 3

Una vez que tenemos las tasas de error sobre training, pasamos a obtener las tasas de error sobre test en algunos casos. Ya que la base de datos SVD posee más componentes e información nos centraremos en ella para este apartado, realizando las mismas particiones que con las GMM. A su vez, como hemos visto que el clasificador con el vector 3 como entrada no ha conseguido tener mejor rendimiento que teniendo el vector 2 como entrada, descartaremos hacer el test, ya que no sería viable realizarla si no supone una mejoría.

En la Tabla 7, podemos ver las tasas de error sobre test y sobre training tras realizar una media de 10 nuevas pruebas con el clasificador y con el vector 1 como datos de entrada. Se puede ver que el clasificador presenta una efectividad mayor al 70% en, prácticamente, todas las particiones de test, por lo que resulta un resultado similar al visto con las GMM.

Partición test	1	2	3	4	5
Tasa de error sobre training	0.2849	0.2836	0.2830	0.2912	0.2836
Tasa de error sobre test	0.2868	0.3020	0.2995	0.2766	0.2893

Tabla 7. Tasa de error sobre test y sobre training con el vector 1

Si usamos el vector 2 como datos de entrada y siendo 16 el número de gaussianas para obtener este vector, en la Tabla 8, podemos ver que el clasificador empeora respecto a la tasa de error sobre training, por lo que, en nuestro caso, no es una buena solución el uso de supervectores para obtener un mayor rendimiento.

Partición test	1	2	3	4	5
Tasa de error sobre training	0.2297	0.2253	0.2214	0.2322	0.2297
Tasa de error sobre test	0.3275	0.3286	0.3282	0.3299	0.3290

Tabla 8. Tasa de error sobre test y sobre training con el vector 2

Para concluir este apartado, hay que destacar que este clasificador, al igual que ocurría con el clasificador con GMM, presenta un clara mejoría en la tasa de error sobre training cuanto mayor sea el número de gaussianas. Pero, sobre la tasa de error sobre test, prácticamente, es similar el porcentaje de acierto entre los dos clasificadores.

4.3. Redes neuronales

El último clasificador utilizado se basa en redes neuronales. En este caso se usarán diferentes datos de entrada a la red para obtener la tasa de error sobre training y compararemos ciertos resultados con uno de los artículos comentados en el apartado de Estado del arte [5], el cual usa la base de datos SVD con el clasificador SVM.

Para empezar esta sección, vamos a ver el procedimiento seguido, de forma general, para diseñar las redes hasta obtener la tasa de error sobre training:

1. Datos de entrada: primero se obtienen los datos de entrada y sus respectivas etiquetas.
2. Diseñar red: usando la función de Matlab 'patternnet' ajustamos el número de capas y los coeficientes de salida de estas para obtener una red de reconocimiento de patrones. También se puede indicar el número de epoch, el número máximo de fallos de la red, etc.
3. Entrenamiento de la red: con la ayuda de la función de Matlab 'train' entrenamos la red que hemos diseñado antes. Y, una vez entrenada, obtenemos la salida de la red, que será una matriz de dos filas y tantas columnas como parámetros de entrada tengamos, con las probabilidades de pertenecer a voz patológica (primera fila) o a no patológica (segunda fila).
4. Tasa de error sobre training: usaremos la salida de la red para obtener la predicción de las etiquetas y, de la misma forma que con el clasificador SVM, se podrá obtener la tasa de error.

Una vez visto la forma para diseñar las redes, pasamos a ver los resultados para los siguientes datos de entrada. En este caso, nos centramos más en utilizar la base de datos SVD, ya que tenemos más información y más datos para entrenar la red. Sin embargo, lo primero, vamos a ver el uso, como datos de entrada, de las medias y desviaciones estándar de los coeficientes MFCC y de los parámetros del OpenSMILE con el estándar eGeMAPS con la base de datos VOICED. En la Tabla 9 podemos ver los valores obtenidos de las tasas de error para diferentes capas de la red. Estos valores se han obtenido al realizar un media de 50 iteraciones con la red para obtener un resultado más globalizado. Como vemos el aumento de las capas en la red neuronal no presenta muchas mejorías, ya que se mantiene una tasa de error sobre training del 25% aproximadamente en todos los casos.

Dimensión de la red	[50]	[100 25]	[100 50 25]	[100 75 50 25]
MFCC	0.2436	0.2412	0.2720	0.2572
eGeMAPS	0.2630	0.2446	0.2314	0.2356

Tabla 9. Tasa de error sobre training con diferentes datos de entrada con la base de datos VOICED

Tras ver unos datos de entrada con una base de datos con pocas componentes, vamos a usar la base de datos SVD para ver si cambiando los datos de entrada se observa un mejor rendimiento al tener más información.

En la Tabla 10, se pueden ver los valores de las tasas de error sobre training tras 50 iteraciones, como hemos hecho antes. Al igual que pasaba con la base de datos VOICED, el aumento del número de capas no cambia la tasa prácticamente, pero si lo hacen los datos de entrada. Se observa que, entre todos los conjuntos de características, el conjunto de características ComParE es el que muestra mejor rendimiento con respecto a los demás. Se obtiene un porcentaje de acierto de un 80%, aproximadamente, cuando en los demás casos

ronda entre el 76% y el 71%. El rendimiento de las características de entonación es el más bajo de todos, siendo casi más bajo que el mínimo teórico, ya que si el clasificador decide que todas las muestras son patológicas se obtiene una tasa de error del 33.74%. Es decir, que con el conjunto de características de entonación el clasificador predice que casi todas las muestras son patológicas.

Dimensión de la red	[50]	[100 25]	[100 50 25]	[100 75 50 25]
ComParE	0.1938	0.1912	0.2160	0.2170
eGeMAPS	0.2420	0.2433	0.2450	0.2518
MFCC	0.2704	0.2744	0.2704	0.2636
Glotal	0.2928	0.2882	0.2882	0.2868
Entonación	0.3370	0.3370	0.3370	0.3370

Tabla 10. Tasa de error sobre training con diferentes datos de entrada con la base de datos SVD

Una vez vista las tasas de error sobre training, pasamos a ver las tasas de error sobre test. Como hemos visto, el número de capas no produce un aumento de la eficiencia, entonces vamos a utilizar una dimensión intermedia ([100 25]) para realizar nuevas pruebas. Al igual que hemos hecho con el clasificador SVM, utilizamos las mismas particiones de la base de datos SVD como entrada. En la Tabla 11, se pueden ver las tasas de error sobre test y sobre training con diferentes datos de entrada tras realizar una media de 10 pruebas. Vemos que sobre training sigue teniendo mejor rendimiento con el conjunto de características ComParE, pero sobre test hay una cierta similitud entre todos los conjuntos de características salvo las características de entonación que sigue siendo el que peor rendimiento presenta.

Test	1	2	3	4	5
ComParE	Test: 0.263 Training:0.183	Test: 0.323 Training:0.188	Test: 0.278 Training:0.162	Test: 0.274 Training:0.186	Test: 0.264 Training:0.197
eGeMAPS	Test: 0.266 Training:0.249	Test: 0.321 Training:0.246	Test: 0.286 Training:0.226	Test: 0.2267 Training:0.254	Test: 0.277 Training:0.224
MFCC	Test: 0.293 Training:0.268	Test: 0.301 Training:0.265	Test: 0.316 Training:0.245	Test: 0.288 Training:0.254	Test: 0.279 Training:0.265
Glotal	Test: 0.297 Training:0.278	Test: 0.334 Training:0.261	Test: 0.324 Training:0.262	Test: 0.302 Training:0.289	Test: 0.302 Training:0.280
Entonación	Test: 0.345 Training:0.337	Test: 0.345 Training:0.337	Test: 0.345 Training:0.337	Test: 0.345 Training:0.337	Test: 0.345 Training:0.337

Tabla 11. Tasa de error sobre test y training con diferentes datos de entrada con la base de datos SVD

Si hacemos una comparación con los resultados obtenidos en el artículo [5], en el que se utilizan los mismos datos que en nuestros experimentos, vemos que se asemejan bastante a los

conseguidos en este trabajo. La única pega que tenemos es que no se sabe si los datos de precisión de clasificación (Tabla 12) de este artículo son los mejores que han obtenido o son una media de todas las pruebas que han realizado. A parte de esto, vemos que hemos conseguido unos resultados similares, donde el conjunto de características ComParE sigue siendo el mejor y el conjunto de características de entonación da también el peor resultado.

Datos de entrada	ComParE	eGeMAPS	MFCC	Glotal	Entonación
Precisión de clasificación (%)	82.8	76.0	74.4	67.4	69.3

Tabla 12. Precisión de clasificación del artículo [5]

En conclusión, la importancia de los datos de entrada es esencial para entrenar redes neuronales, ya que cuanta más información tengan mejor rendimiento presentan. Un ejemplo de esto es el conjunto de características ComParE, ya que tiene el que más información de todos los tipos de datos de entrada. Hay que destacar que el cambio del número de capas en las redes, en este caso, no ha hecho que se aumente la eficiencia.

5. CONCLUSIONES

En este trabajo, se ha visto que las técnicas de aprendizaje automático junto a la parametrización de señales de voz ofrecen una gran posibilidad en el tratamiento de este tipo de señales.

Se han obtenido diferentes parámetros frecuenciales (MFCC), acústicos (OpenSMILE) y temporales (glotal) que ofrecen cierta información de las bases de datos que hemos utilizado (VOICED y SVD) para poner a prueba a los tres clasificadores (GMM, SVM, Red neuronal).

Gracias a los clasificadores GMM y SVM, se llega a la conclusión de la importancia de la elección de las bases de datos, ya que cuanto más grande sea la base de datos, el clasificador tendrá más información para clasificar las señales de entrada y, por tanto, más rendimiento. Además, el número de gaussianas tiene que ser adecuado para la cantidad de información que tenemos, ya que cuanto más información tenemos, necesitamos más gaussianas para obtener una mayor precisión.

A parte de la importancia de la elección de las bases de datos, la parametrización también es esencial para la clasificación, sobre todo en redes neuronales. Se ha comprobado que no todos los conjuntos de parámetros obtenidos son importantes a la hora de realizar la clasificación. El conjunto de característica del estándar ComParE queda por encima de los demás teniendo casi un 80% de efectividad sobre el entrenamiento, mientras que los parámetros de entonación no llegaban ni al 70%.

Por último, hay que destacar la importancia de estas técnicas de aprendizaje automático para el ámbito sanitario y la evolución tecnológica que han experimentado. Gracias a ellas, se pueden suprimir ciertas técnicas invasivas que se realizaban a los pacientes para detectar patologías y hacer el proceso más rentable y eficiente.

6. REFERENCIAS

[1] S. Jothilakshmi. *Automatic system detect the type of voice pathology*, 2014. Applied Soft Computing Volume 21, August 2014, Pages 244-249

[2] Sarika Hegde, Surendra Shetty, Smitha Rai, Thejaswi Dodderi. *A survey on machine learning approaches for automatic detection of voice disorders*, 2018. Journal of Voice, Volume 33, Issue 6, November 2019, Pages 947.e11-947.e33.

[3] Mark Huckvale, Catinca Buciuleac. *Automated detection of voice disorder in the Saabrücken Voice Database: effects of pathology subset and audio material*, 2021. INTERSPEECH 2021. https://www.isca-speech.org/archive/pdfs/interspeech_2021/huckvale21_interspeech.pdf

[4] Ugo Cesari, Giuseppe De Pietro, Elio Marciano, Ciro Niri, Giovanna Sannino, Laura Verde. *A new database of heatlhy and pathological voices*, 2018. Computers &Electrical Engineering, Volumen 68, páginas 310-321.

[5] Purva Barche, Krishna Gurugubelli, Kumar Vuppala. *Towards automatic assessment of voice disorders: a clinical approach*, 2020. Interspeech 2020, 2537-2541, doi: 10.21437/Interspeech.2020-2160.

[6] Björn Schuller, Stefan Steidl, Anton Batliner, Julia Hirschberg, Judee K. Burgoon, Alice Baird, Aaron Elkins, Yue Zhang, Eduardo Coutinho, Keelan Evanini. *The INTERSPEECH 2016 Computational Paralinguistics Challenge: Deception, Sincerity & Native Language*, 2016. Presentado en: 17th Annual Conference of the International Speech Communication Association, INTERSPEECH 2016, San Francisco, United States.

[7] Florian Eyben, Klaus Scherer, Björn Schuller, JohanSundberg, Elisabeth André, Carlos Busso, Laurence Devillers, Julien Epps, Petri Laukka, Shrikanth Narayanan, Khiet Truong. *The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing*, 2015. Obtenido de: IEEE transactions on affective computing, Vol.7, No.2, 04.2016, p. 190-202.

[8] Jonathan Hui. *Speech Recognition – Feature Extraction MFCC & PLP*, 2019. Obtenido de: https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9

[9] Namrata Dave. *Feature Extraction Methods LPC, PLP and MFCC in Speech Recognition*, 2013. Obtenido de: https://www.researchgate.net/publication/261914482_Feature_extraction_methods_LPC_PLP_and_MFCC_in_speech_recognition

[10] Wikipedia (sin fecha) Obtenido de: https://en.wikipedia.org/wiki/Mel_scale

[11] Ludovic Rembert. *Mel Frequency Cepstral Coefficient*, 2021. Obtenido de: https://privacycanada.net/mel-frequency-cepstral-coefficient/

[12] J. McGonagle, G. Pilling, A. Dobre, V. Tembo, A. Kurmukov, A. Chumbley, E. Ross, J. Khim. *Gaussian Mixture Model*, 2021. Obtenido de: https://brilliant.org/wiki/gaussian-mixture-model/

[13] Wikipedia (sin fecha) Obtenido de: https://en.wikipedia.org/wiki/Support-vector_machine

[14] Arthur Arnx. *First neural network for beginners explained*, 2019. Obtenido de: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf

[15] MathWorks (sin fecha). *¿Qué es una red neuronal?* Obtenido de: https://es.mathworks.com/discovery/neural-network.html?s_tid=srchtitle_neural%20network_1

[16] Nick McCullum. *Deep Learning Neural Networks Explained in Plain English*, 2020. Obtenido de: https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/

7. ANEXOS

En este apartado, vamos a ver tres scripts de Matlab como ejemplo de programas utilizados en este trabajo para obtener los tres clasificadores.

7.1. GMM

```
% ejemplo de clasificación patología/no patologiacon GMM con VOICEd
patfilegui="../listas/pathologicalfiles.lst";  % fichero guia con los ficheros de voz patológica
nopatfile="../listas/healthyfiles.lst"; % fichero guia con los ficheros de voz sin patologia
% creamos primero los ficheros con los coeficientes mfcc.
% un vector de medias + desviacion standard por fichero
Cpat=meanmfccCreate(patfilegui,23);
Cnpat=meanmfccCreate(nopatfile,23);
%%
% entrenamos el modelo gmm
K=32;  % número de gaussianas
% modelo voz con patología
fileout="/Users/alvar/Desktop/TFG_Matlab/Matlab/Data/ubm/meangmm"+int2str(K)+"-pathologicalfiles.mat";
patgmm=fitgmdist(Cpat,K,'CovarianceType','diagonal','SharedCovariance',true,'Options',statset('Display',
'iter','MaxIter',5000),'RegularizationValue',0.001);
save(fileout,'patgmm');  % guardamos el modelo en un fichero
% modelo voz sin patología
fileout="/Users/alvar/Desktop/TFG_Matlab/Matlab/Data/ubm/meangmm"+int2str(K)+"-nopathologicalfiles.mat";
nopatgmm=fitgmdist(Cnpat,K,'CovarianceType','diagonal','SharedCovariance',true,'Options',statset('Display',
'iter','MaxIter',5000),'RegularizationValue',0.001);
save(fileout,'nopatgmm');  % guardamos el modelo en un fichero
```

```
% Vamos a hacer estadísticas con los vectores de entrenamiento
% Primero para la voz con patología
% log(Probabilidad) de cada fichero con voz patológica con el modelo de voz patológica
pdf_pat=log(pdf(patgmm, Cpat));
% log(Probabilidad) de cada fichero con voz patológica con el modelo de voz no patológica
pdf_no_pat=log(pdf(nopatgmm, Cpat));
% Ratio de verosimilitud para la voz con patología
lrp=pdf_pat-pdf_no_pat;
%%
% Vamos a hacer estadísticas con los vectores de entrenamiento
% SEgundo para la voz sin patología
% log(Probabilidad) de cada fichero sin voz patológica con el modelo de voz no patológica
pdf_no_pat=log(pdf(nopatgmm, Cnpat));
% log(Probabilidad) de cada fichero sin voz patológica con el modelo de voz  patológica
pdf_pat=log(pdf(patgmm, Cnpat));
% Ratio de verosimilitud para la voz no patológica
lrnp=pdf_pat-pdf_no_pat;
```

```
%%
% Vamos a dibujar la curva ROC, calcular el área bajo la curva (AUC) y el punto óptimo de trabajo
% Creamos un vector de etiquetas:  1 patológica, 0 no patológica
R=[ones(size(lrp));zeros(size(lrnp))];
% Creamos un vector con los ratios de verosimilitud correspondientes a las etiquetas
scores=[lrp;lrnp];
% Utiliamos la función perfcurve de matlaba para dibujar la curva ROC y calcular AUC y punto óptimo
[X,Y,T,AUC,OPTROC] = perfcurve(R,scores,1);
plot(X,Y)
text=sprintf("AUC: %f, OPTROC: %f, %f",AUC,OPTROC);
title(text)
disp(text)
```

7.2. SVM

```
% ejemplo de entrenamiento de un ubm con VOICEd
filegui="../listas/allfiles.lst";
outpath="./mfcc/";
% creamos primero los ficheros con los coeficientes mfcc.
mfccCreate(filegui,outpath,23);
Datameansv=meanmfccCreate(filegui,23);   % para entrenar svm solo con media + sd
labels=readlabelsbinary("../listas/allfilesgroupdesease.lst");
%%
% calculamos un clasificador SVM solo con el vector de media+sd
labels=readlabelsbinary("../listas/allfilesgroupdesease.lst");
SVMModelms = fitcsvm(Datameansv,labels);
CVSVMModelms = crossval(SVMModelms);
classLossms = kfoldLoss(CVSVMModelms);
fprintf(1,"Tasa de error (10-fold cross-validation: %f\n",classLossms);
[labelms,scorems] = predict(SVMModelms,Datameansv);
error=sum(abs(labels-labelms))/length(labels);
fprintf(1,"Tasa de error sobre training: %f\n",error);
%%
% Creamos el train set. Se trata de una matriz con todas las tramas para
% entrenar el UBM
% Estructura:
%   fila 1: vector de medias
%   fila 2: vector de desviacion estandar
%   resto de filas los vectores de mfcc
%
filedat="../listas/allfilesmfcc23.lst";
createTrainset(filedat,"./Data/allfilesTrainmfcc23.dat");
```

```
%%
% Leemos la matriz con todas las tramas para estimar el UBM y normalizamos
% a media cero y varianza unidad cada coeficiente mel-cepstrum.
data=readmatrix("./Data/allfilesTrainmfcc23.dat");
datamedia=data(1,:);
datastd=data(2,:);
Data=(data(3:end,:)-datamedia)./datastd;
%clear data;
%%
% entrenamos el ubm
K=16;  % número de gaussianas
fcoef=1; % primer coeff cepstrum a utilizar
lcoef=23; % último coeff cepsturm a utilizar
ncoef=lcoef-fcoef+1;
fileout="./Data/ubm/gmm"+int2str(K)+"-allfiles.mat";
ubm=fitgmdist(Data(:,fcoef:lcoef),K,'CovarianceType','diagonal','SharedCovariance',true,'Options',
statset('Display','iter','MaxIter',5000),'RegularizationValue',0.001);
save(fileout,'ubm');  % guardamos el modelo en un fichero
%%
% vamos utilizar adaptación MAP para crear supervectores para cada fichero
r=16; % factor relevancia adaptacón MAP
iter=1; % número iteraciones MAP
SV=createSV(filedat,ubm,r,iter,fcoef,lcoef,datamedia,datastd);
```

```
%%
% calculamos un clasificador SVM

SVMModel = fitcsvm(SV',labels);
CVSVMModel = crossval(SVMModel);
classLoss = kfoldLoss(CVSVMModel);
fprintf(1,"Tasa de error SV(10-fold cross-validation: %f\n",classLoss);
[label,score] = predict(SVMModel,SV');
error=sum(abs(labels-label))/length(labels);
fprintf(1,"Tasa de error sobre training: %f\n",error);
```

```
%%
% representamos los supervectores con tsne
figure
Y=tsne(SV','Algorithm','exact','Distance','euclidean');
labelsd=readlabels("../listas/allfilesgroupdesease.lst");
gscatter(Y(:,1),Y(:,2),labelsd)
```

```
%%
% representación de toda la base de datos VOICED
%figure
%YD=tsne(Data(:,fcoef:lcoef));
%gscatter(YD(:,1),YD(:,2))
```

```
%  Ejemplo RSR
lambda=0.7;
[err,W,E]=RSR(SV',lambda);
figure
wc=sum(W.*W,2);
plot(wc);
figure
plot(E)
```

```
%%
%
i=find(wc>0.8);
WR=W(:,i);
size(WR)
SVR=SV'*WR;
```

```
%%
% calculamos un clasificador SVM
labels=readlabelsbinary("../listas/allfilesgroupdesease.lst");
SVMModelR = fitcsvm(SVR,labels)
CVSVMModelR = crossval(SVMModelR);
classLossR = kfoldLoss(CVSVMModelR);
fprintf(1,"Tasa de error RSR, k= %d (10-fold cross-validation: %f\n",length(i),classLossR);
[labelSV,scoreSV] = predict(SVMModelR,SVR);
error=sum(abs(labels-labelSV))/length(labels);
fprintf(1,"Tasa de error sobre training: %f\n",error);
```

```
%%
% representamos los supervectores con tsne
figure
YR=tsne(SVR,'Algorithm','exact','Distance','euclidean');
labelsd=readlabels("../listas/allfilesgroupdesease.lst");
gscatter(YR(:,1),YR(:,2),labelsd)
```

7.3. Redes neuronales

```
%% RED NEURONAL

clear all;
close all;
clc;

% creamos primero los ficheros con los coeficientes mfcc.
patfilegui="../listas/pathologicalfiles_SVD.lst";  % fichero guia con los ficheros de voz patológica
nopatfile="../listas/healthyfiles_SVD.lst"; % fichero guia con los ficheros de voz sin patologia
% creamos primero los ficheros con los coeficientes mfcc.
% un vector de medias + desviacion standard por fichero
Cpat=meanmfccCreate(patfilegui,23);
Cnpat=meanmfccCreate(nopatfile,23);
C=[Cpat;Cnpat];
labels=[ones(length(Cpat),1);zeros(length(Cnpat),1)];

t=zeros(2,length(C));
t(1,:)=labels;
t(2,:)=1-labels;

%%
net=patternnet([100,75,50,25]);
net.trainParam.epochs=1000;
net.trainParam.max_fail=100;
net.trainParam.min_grad=1e-29;
%net.trainParam.mu=0.1;
%net.trainParam.mu_dec=0.1;
%net.trainParam.mu_inc=10;

for j=1:10

    net=train(net,C',t);
    %view(net);
    y=net(C');
    perf(j)=perform(net,y,t);

    ind=vec2ind(y);
    i=find(ind==2);
    label=ind;
    label(i)=0;
    error(j)=sum(abs(labels'-label))/length(labels);
    fprintf(1,"Tasa de error sobre training: %f\n",error(j));

    muestras_erroneas(:,j)=labels'~=label;
end

error_total=sum(abs(error))/length(error);
fprintf(1,"Media de la tasa de error sobre training: %f\n",error_total);
```