Trabajo Fin de Grado
Grado Ingeniería de las Tecnologías de las Telecomunicaciones

Clasificación de géneros musicales mediante técnicas de aprendizaje automático

Autor: Miguel Pérez Fernández

Tutoras: Maria Auxiliadora Sarmiento Vega, Irene Fondón García

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Departamento de Teoría de la Señal y Comunicaciones

Trabajo Fin de Grado

Grado Ingeniería de las Tecnologías de las Telecomunicaciones

Clasificación de géneros musicales mediante técnicas de aprendizaje automático

Autor:

Miguel Pérez Fernández

Tutoras:

Maria Auxiliadora Sarmiento Vega, Irene Fondón García

Profesora contratada doctora interina, Profesora contratada doctora

Dpto. Teoría de la Señal y Comunicaciones

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado: Clasificación de géneros musicales mediante técnicas de aprendizaje automático

Autor: Miguel Pérez Fernández
Tutoras: Maria Auxiliadora Sarmiento Vega, Irene Fondón García

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Quiero aprovechar estas lineas para agradecer a mi familia el apoyo que me han brindado toda mi vida, a mis tutoras por ayudarme a avanzar y a la vida por enseñarme la música.

Miguel Pérez Fernández

Sevilla, 2018

Resumen

La música ha formado parte fundamental de la humanidad desde el inicio de sus tiempos, tanto como por simple ocio, como ejercicio artístico e intelectual y como cohesión para vínculos sociales. Conforme los tiempos avanzaron así lo hicieron las las tecnologías y la forma en la que la gente se relacionaba con los elementos que ya había presente con anterioridad a estas. La capacidad de alcance de música que en principio pertenecían a una región ahora se había multiplicado, y no solo eso, también aumentó la facilidad con la que puede consumirse música con elementos como Spotify, Deezer, Shazam… No obstante cómo funciona exactamente la música y que procesos cognitivos suceden en el cerebro mientras la escuchamos (o recordamos) sigue siendo un misterio. Tanto es así que existen incluso organizaciones y "challenges" como el MIREX que se dedican a promover su investigación desde un punto de vista ingenieril. Una de las tareas que suscita mas interés es la de clasificación de los géneros musicales, el qué es lo que ayuda a definir cada forma de música. Aquí nos centraremos en 5 géneros en entorno a la música *latina*: Salsa, Bachata, Reggaeton, Merengue y Chachachá.

Índice Abreviado

Índice

1 Introducción y objetivos

Sin música la vida sería un error.

Friedrich Wilhelm Nietzsche

La tarea de la clasificación de géneros musicales no es nueva, aunque no por ello es de las más avanzadas en materia de análisis musical con métodos computacionales. En este capítulo hablaremos del contexto en el que se desarolla esta materia y qué pretendemos llegar a obtener con este trabajo.

1.1 Contexto

Spotify nace en 2006 [1] como una empresa de *streaming* de música, no obstante no fue la pionera en este tipo de servicios, los primeros servicios de *streaming* de música en internet apuntan a *pandora* y *Last.fm* como primeras incursoras. Lo que sí hizo Spotify fue acercar la música al público masivo simplificando la búsqueda, la selección y la clasificación de los artistas, estilos, álbumes, ... Y es que las dos otras empresas mencionadas ofrecían interfaces muy completas, pero muy complejas. Desde la llegada de internet la cantidad de información disponible y las estructuras que forma se han complicado de manera exponencial, sin embargo se ha tendido a simplificarlo todo mucho más puesto que la intención del ocio es el consumo. De esta necesidad de simplificar nació *shazam* la cual permite a cualquier usuario saber qué esta escuchando al instante. Y no solo el usuario individual es el objetivo final de estas tecnologías, trabajos como el de *DJ* comienzan a ver como peligran sus puestos de trabajo con las herramientas de "autopinchado" [2]. No obstante el proceso y tarea de catalogación de la música si que sigue siendo mayormente manual, aunque la música se puede clasificar de muchas maneras, por géneros, emociones, propias de una región o no, etc... Una tarea que está vigente hoy en día y en la que en este trabajo pretendemos ahondar.

1.2 Objetivos

Nuestro objetivo con este trabajo es estudiar la clasificación de 5 de géneros de musica latina en base a su relevancia en el panorama popular actual:

- Reggaeton
- Salsa
- Bachata
- Merengue
- Chachachá

El reggaetón es un género nacido en los 90 en Puerto Rico que aunque tiene una fuerte influencia del *hip-hop*, está mayormente derivado del *reggae*, de hecho toma a este género como prefijo de su propio nombre, *reggae*-tón. Aprovechando la popularidad de la que disfrutaba el *reggae* jamaicano a principios de los 90 se introduce el *reggae* panameño en español y es cuando se empieza a desarrollar el género que conocemos hoy día, prueba de ello es que el que el *Dancehall* (una variante del *reggae*) comparte la misma base rítmica que tanto caracteriza al reggaetón. En general se caracteriza por un *BPM* (*beats per minute*, o pulsos por minuto) relativamente bajo con respecto a las otras clases señaladas y un ritmo muy característico. Además por su naturaleza de música nacida en los estudios, la mayoría de instrumentos son sintetizados, desembocando en que las características espectrales pueden variar abruptamente entre canciones, especialmente a lo relativo a los instrumentos que no funcionan como bajo en la canción.

La salsa es un género musical resultante del son cubano y otros géneros de música caribeña. No obstante su éxito comercial aparece alrededor de 1960 gracias a músicos puertorriqueños en Nueva York, aunque sus raíces se remontan a décadas anteriores en países de la cuenca del Caribe [3]. Salsa realmente es un término paraguas para lo que en realidad son varios géneros de la música cubana: mambo, guarachá, chachachá, guaguancó citetitoentrevista, etc, aunque la espina dorsal es el son cubano. El ritmo de la salsa utiliza como base la clave del son, que puede ser del tipo 2-3 o 3-2, tiene un tempo muy elevado aunque los estimadores de tempo siempre detectan su tempo a la mitad de velocidad (se verá mas adelante el porqué en el capítulo de características). Apenas usa instrumentos sintetizados y los instrumentos mas importantes suelen ser los percusivos pues son quienes suelen guiar a los bailarines a no perder el ritmo en la pista de baile: timbales, bongó, cencerro, maracas y conga entre otros. Para las melodías se suelen usar orquestas compuestas por trompetas, saxofón, piano, contrabajo o bajo eléctrico, trombones, flauta y violín. Esto en cierto modo puede fijar, o no dar tanta libertad a la forma que adquiere el espectro de frecuencias.

La bachata es un género musical originario de la República Dominicana dentro del folclore urbano. Está asociada a la marginalidad urbana de los bares y burdeles de Santo Domingo durante finales de los años 60 y principios de los años 70, ganándose la fama de “música de amargue”, puesto que la mayoría de canciones tratan sobre el desamor. La etimología en cambio lleva a su origen africano que designa juerga, jolgorio y parranda [4].

Este género es de los que más se ha renovado con el tiempo, en el sentido en que a pesar haber nacido en los tiempos de los orígenes de la salsa, chachachá, etc, ha cambiado tanto su tempo como los instrumentos de los que se compone, relegando por ejemplo la guitarra que solía acompañar sus melodías por otros instrumentos sintetizados, pero manteniendo similar su base rítmica.

El merengue es mas longevo de todos estos géneros, tiene sus raíces a finales del siglo XIX en República Dominicana, aunque su popularidad está extendida por toda Sudamérica y de hecho es Patrimonio Cultural Inmaterial de la Humanidad según la Unesco [5]. En sus orígenes era interpretado con instrumentos de cuerda (bandurria y/o guitarra), aunque con el tiempo se fueron susituyendo por acordeón, guira y la tambora. En 1875 el presidente Ulises Francisco Espaillat inicia una campaña contra el merengue por sus bailes y letras explícitas [6], en vano, pues para entonces la música ya se había adueñado de Cibao. Desarrolló dos versiones, una que se tocaba mas en el campo y guardaba las tradiciones mas antiguas del merengue, y otra que se denominó merengue de salón, propio de los centros urbanos. Al igual que la salsa es un género muy rápido y con una instrumentación muy bien definida, no obstante aquí los primeros tiempos están muy bien marcados y definidos, por lo que no sufre el problema que existe en la salsa a la hora de estimar el tempo.

El chachachá es un género de la música panameña desarrollado a partir del danzón-mambo a comienzos de los 50 [7] y se suele atribuir su origen al compositor y violinista cubano Enrique Jorrín. El nombre toma su origen en el sonido que suelen hacer los bailarines con los pies al acoplarse el baile con la música, como una onomatopeya que suena “chachachá”. El principal elemento que lo diferencia es su célula rítmica, que a diferencia de por ejemplo la célula ritmica de la clave en la salsa, solo posee una variante, no dos. Es una música relativamente lenta y no suele variar su instrumentación, al igual que el merengue o la salsa.

Para la tarea de clasificación de estos géneros mencionados utilizaremos varios algoritmos de *machine learning*, *supervised* y *unsupervised*(más adelante explicaremos en que consiste cada uno de ellos en el apartado de aprendizaje de máquinas). Haremos un estudio probando el *clustering* mediante *k-means* (para el caso de los *unsupervised*) con el fin de comprobar como podría funcionar este sistema con los géneros a ciegas, sin etiquetas. No obstante probaremos con varios métodos de *supervised learning* con cuales se suelen obtener mejores resultados y sopesaremos cual sería mas factible de cara a la implementación de un sistema funcional.

En la figuras 1.5, 1.1, 1.4, 1.3 y 1.2 podemos ver distintos ejemplos de espectros para los distintos géneros de música.

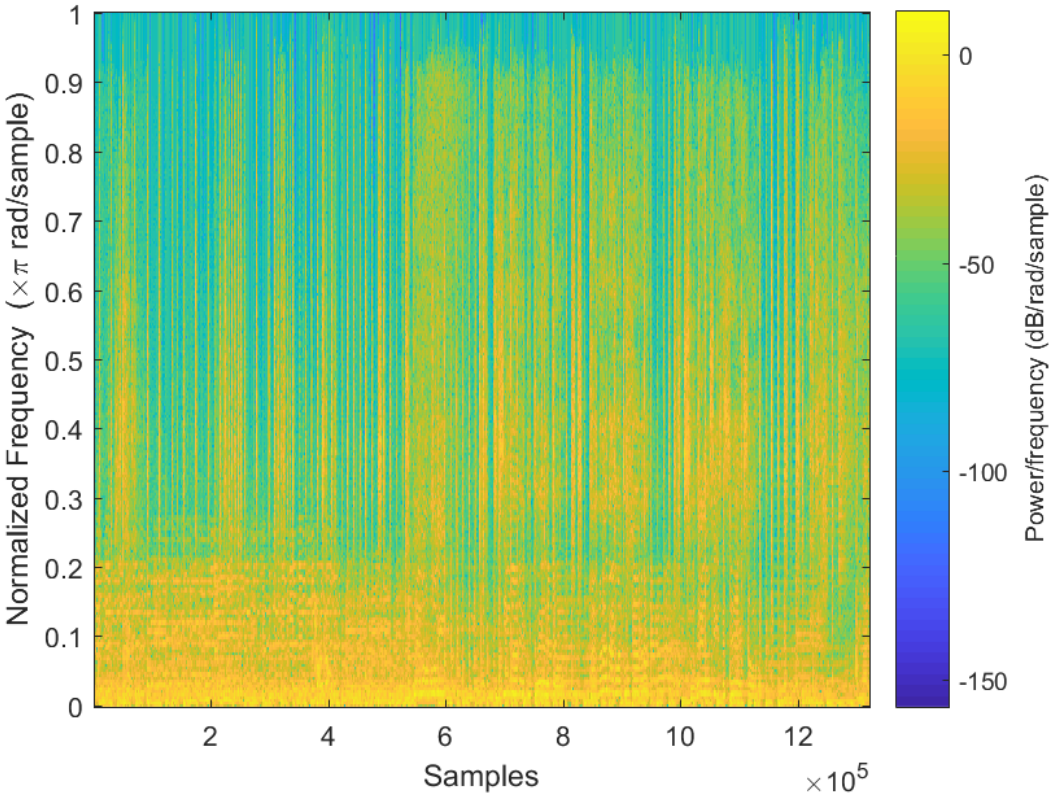

Figura 1.1 Espectrograma para bachata.

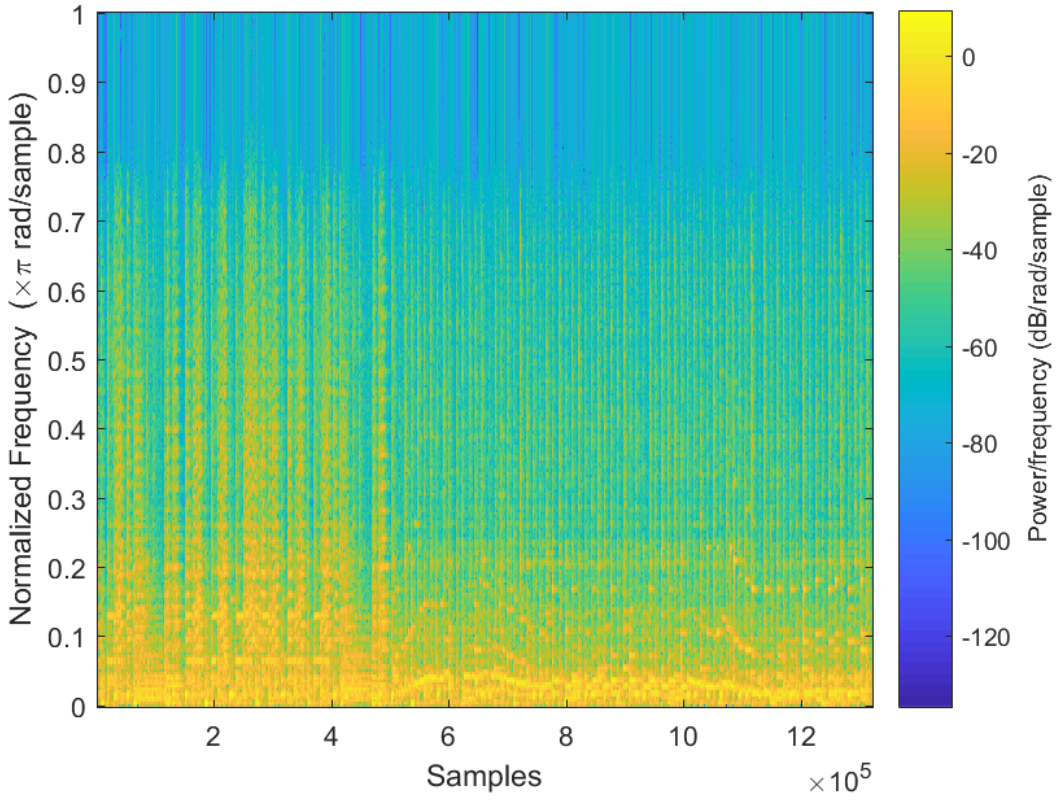

Figura 1.2 Espectrograma para chachachá.

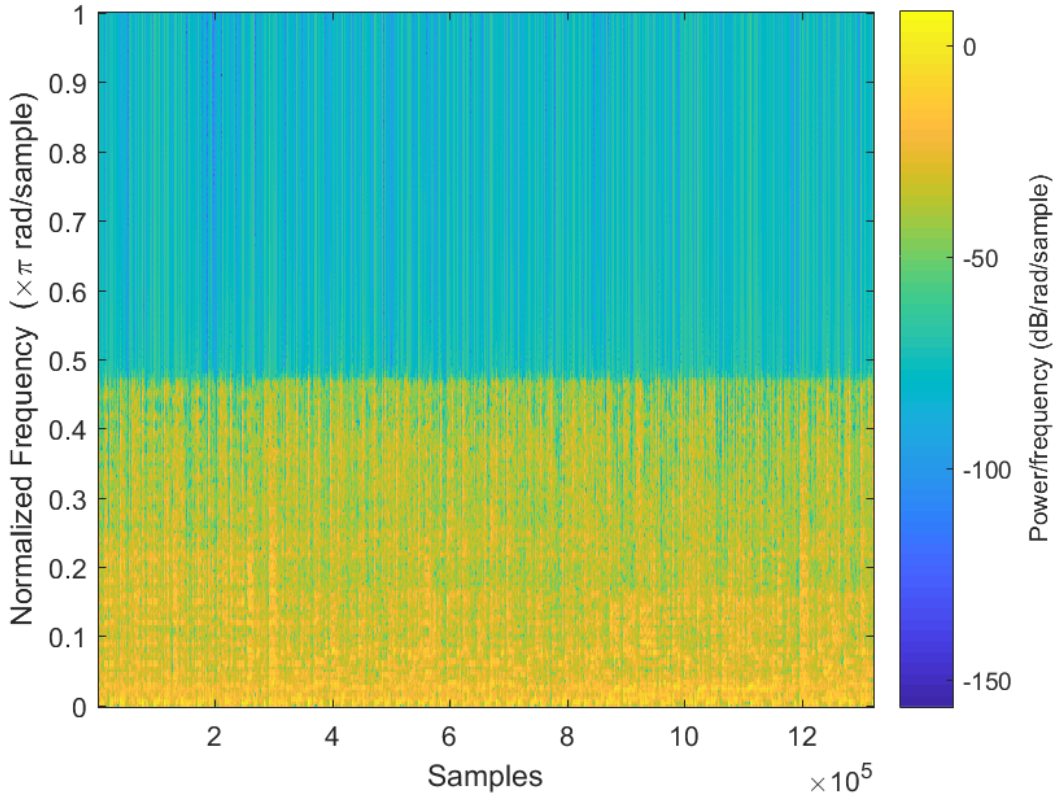

Figura 1.3 Espectrograma para merengue.

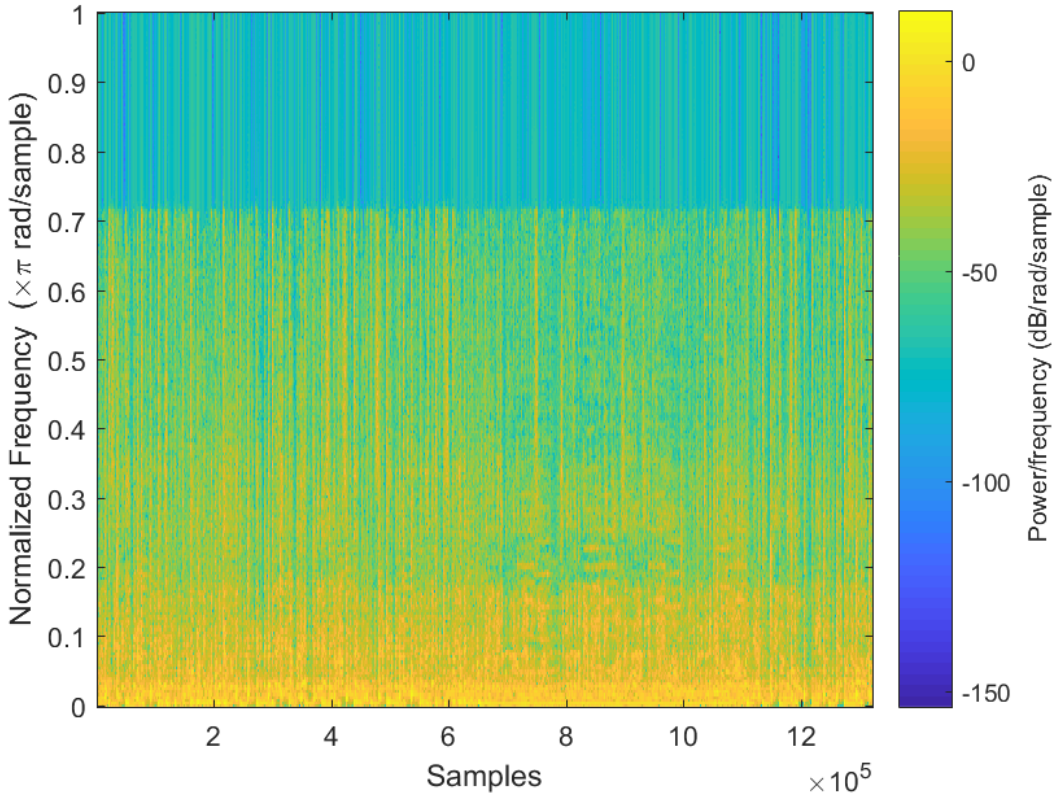

Figura 1.4 Espectrograma para reggaeton.

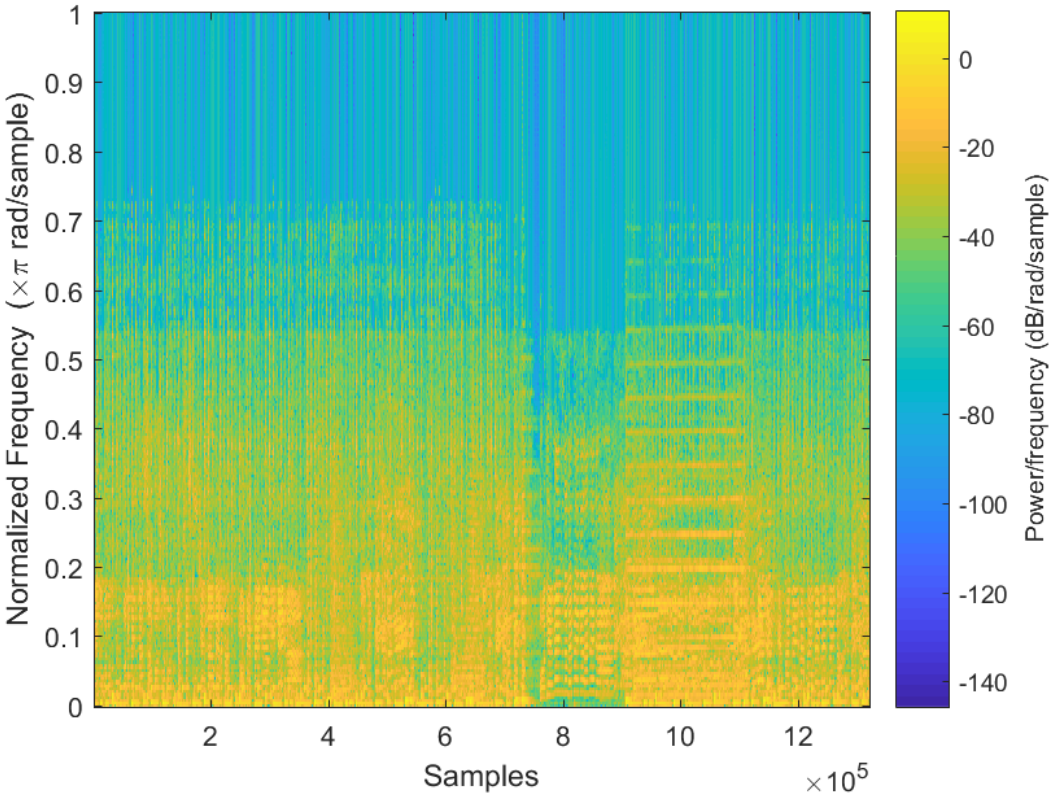

Figura 1.5 Espectrograma para salsa.

2 Estado del arte

Cuando la música agrada al oído y calma el coraón y los sentidos, entonces la música ha perdido el sentido

Maria Callas

En esta sección vamos a hacer un recorrido por los métodos mas avanzados hasta ahora existentes con el mismo propósito que este trabajo de fin de grado.

MIREX es la abreviatura de *Music Information Retrieval Evaluation eXchange* un challenge que se celebra de manera anual como parte del ISMIR (*International Society for Music Information Retrieval Conference*) para promover el estudio de tareas relacionadas con la extracción de datos musicales. Cada año se presentan una serie de tareas o *tasks* para que pueda ser un intercambio de información mucho mas plausible. Por ejemplo las existentes esta edición de 2018 [8] son entre otras:

- Audio Classification (Train/Test) Tasks:
  - Audio US Pop Genre Classification
  - Audio Latin Genre Classification
  - Audio Music Mood Classification
  - Audio Classical Composer Identification
- Audio K-POP Mood Classification
- Audio K-POP Genre Classification
- Audio Fingerprinting
- Audio Onset Detection

- Audio Key Detection
- Audio Chord Estimation
- ...

La lista es larga y variada, desde la clasificación de géneros musicales a hasta la extracción de *onsets* o acentos, pasando por la estimación de acordes o la clave en la que se encuentra la pieza analizada. Todos los resultados son después recogidos en la base de datos de la página del MIREX para estudios posteriores. Un ejemplo de resultados para las tareas señaladas los tenemos en la figura 2.1.

	Composer	Latin	Mood	Mixed K –Genre by K		K-Genre by A	K-Mood by K	K-Mood by A
SubID ▾	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
	(2011) 0.79	(2011) 0.8231	(2017) 0.6983	(2014) 0.8355	(2014) 0.8355	(2017) 0.6774	(2017) 0.6534	(2017) 0.6534
LPNKK1	0.4307	**0.7586**	0.6783	**0.7684**	0.6753	0.669	**0.6534**	0.6458
LPNKK2	0.4159	0.6511	0.665	0.7113	0.6758	0.6621	0.6409	0.6521
LPNKK3	0.4311	0.7347	0.6867	0.7679	**0.679**	**0.6774**	0.6479	**0.6534**
PLNPH1	0.4852	0.6619	**0.6983**	0.7427	0.6521	0.6389	0.6096	0.6277
WJ1	---------	---------	---------	0.7627	---------	---------	---------	---------
XLJ1	**0.6017**	0.6148	0.62	0.6790	0.6114	0.5834	0.5971	0.6207
								Task Captain: Yun Hao (IMIRSEL)

Figura 2.1 Resultados de las tareas de clasificación del MIREX 2017, en las columnas están las distintas tareas de clasificación y en las filas los nombres abreviados de los distintos métodos presentados.

Existen trabajos relativos a la clasificación de géneros musicales usando técnicas clásicas de aprendizaje de máquina que basan su algoritmo en la extracción de los MFCC *(Mel Frequency Cepstral Coefficients)* [9][10] (se verán mas adelante en el capítulo de características) y unos alternativos MFCC extraidos a partir de la FrFFT *Fractional Fast Fourier Transform*, aunque el primero [10] lo aplica principalmente al reconocimiento de instrumentos y es mas tarde cuando el autor lo utiliza para reconocimiento de géneros musicales en [9]. Otros *papers* usan características de mas alto nivel como la progresión de acordes [11] y las reglas de armonía inducidas de los mismos [12]. El problema de estas es que dependen de características que actualmente son muy complicadas de extraer en música polifónica, por lo que su alto índice de error puede empeorar notablemente los resultados.

Existe además otra serie de técnicas mucho mas actuales, como por ejemplo las redes neuronales, que son las mas que mas éxito han tenido en las tareas de clasificación enviadas al MIREX 2017 [13][14][15] y cuyas tasas de éxito pueden verse en la figura 2.1. Las redes neuronales han supuesto un avance sin precedentes en el mundo del aprendizaje automático [16] aprovechando la gran potencia de cálculo que se está desarollando y la capacidad de cómputo paralelo de arquitecturas como *CUDA (Compute Unified Device Architecture)* [17].

En trabajos como [15] los autores usan redes neuronales convolucionales no solo para clasificar géneros de una tarea del MIREX, sino que son capaces de obtener resultados excelentes (véase la figura 2.1) simultáneamente en otras como clasificación de compositores clásicos, clasificación de géneros de música latina (figuras 2.2 y 2.3), *mood genre clasificación*, clasificación de géneros de musica popular, clasificación de géneros de *KPOP* previamente clasificados por personas coreanas y americanas (dos *datasets* distintos) y *KPOP mood genre clasificación* previamente clasificados por personas coreanas y america-

nas (dos *datasets* distintos). Los resultados son asombrosos, exceptuando en el campo de compositores clásicos el citado trabajo se encuentra entre las puntuaciones más altas para el resto de tareas y es el que obtuvo la mayor puntuación en todas las tareas relacionadas con el *KPOP*, haciendo uso de dos bases de datos:

- *Million Song Dataset with Last.FM tag annotations. [18]* (MSD)
- *NAVER Music with genre annotations.*

Para ello usan características de las tres últimas capas ocultas de otras redes ya entrenadas con el método descrito en [19], las cuales se unirán a un único vector para finalmente realizar la clasificación mediante una máquina de soporte vectorial.

En el caso de [13] también podemos ver la eficacia de la redes neuronales. En este caso se entrena una *Deep convolutional neural network* (DCNN) utilizando las canciones en [18]. Esta utiliza como salida varias neuronas en las cuales hay en cada una una etiqueta de los artistas de la base de datos, así la DCNN extrae los vectores de características a partir de 3 segundos de audio de las canciones, estas características después son pasadas a una máquina de soporte vectorial para realizar las clasificaciones tanto de género como de ánimo, como paso previo al uso de la SVM los vectores de características pasan por un análisis de discriminantes lineales (más adelante se hablará de ellos en 3.3) de forma que mejore la capacidad discriminante del espacio donde se realiza la clasificación.

En [14] realizan un preprocesamiento llevando las muestras al dominio de la frecuencia, mediante un filtrado Mel (Se verá mas adelante en 4.1.1) y finalmente pasándolo a dB usando la librería de *Python Librosa* [20]. Con el fin de tener cierta aleatoriedad en las muestras se toman 120 extractos aleatorios a lo largo del tiempo de cada canción, se concatenan esas muestras y se añaden al set de entrenamiento, de esta forma se aumenta la robustez de este clasificador bajo circunstancias de bases de datos pequeñas (un gran problema en algoritmos de Deep Learning). Tras esto se genera un modelo de tres capas de CNN del que finalmente se obtendrán los resultados.

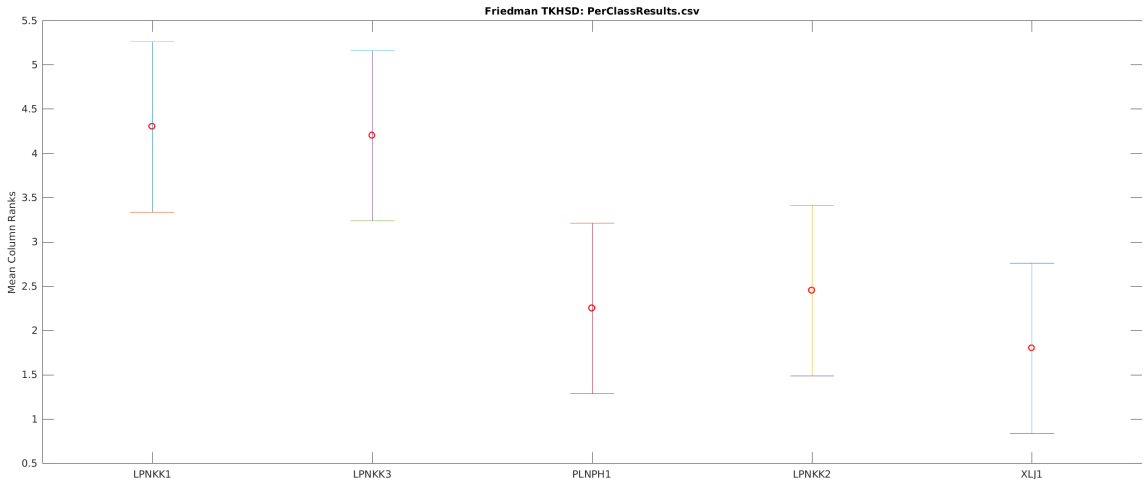

Figura 2.2 Test de Friedman de la precisión por clases para los clasificadores LPNKK1, LPKK3, PLNPH1 [15], PLNPH1 [13] y XLJ1 [14]. Obtenido de [21].

Accuracy per Class

Class	LPNKK1	LPNKK2	LPNKK3	PLNPH1	XLJ1
axe	49.5208	19.4888	43.7700	38.3387	27.1565
bachata	96.8051	93.9297	95.8466	95.5272	92.9712
bolero	83.8095	81.5873	84.4444	74.9206	76.8254
forro	75.7188	64.2173	76.3578	55.5911	50.4792
gaucha	73.0769	55.7692	67.9487	64.1026	49.6795
merengue	96.5079	91.1111	95.5556	95.2381	85.7143
pagode	75.1634	50.6536	67.9739	49.0196	40.1961
salsa	87.7814	84.5659	89.0675	84.5659	78.1350
sertaneja	34.8910	19.9377	27.1028	15.8879	21.4953
tango	83.8235	84.5588	84.3137	84.0686	85.2941

Figura 2.3 Precisión por clases de LPNKK1, LPKK3, PLNPH1 [15], PLNPH1 [13] y XLJ1 [14]. Obtenido de [21].

Sin embargo aunque como ha quedado demostrado en [19] que las redes neuronales funcionan muy bien con respecto a otras técnicas de aprendizaje automático mas clásico, existe un problema con este tipo de técnica. Una red neuronal es una caja negra en el sentido en que aunque puede aproximar cualquier funcion (una función discriminante en este caso), estudiar la estructura de la red entrenada no proporciona ninguna información de la función que está siendo aproximada (aunque hay trabajos para solventar este problema [22]), así como en principio, tampoco proporciona información sobre que características del sistema son verdaderamente importantes [23].

Es por ello que se siguen haciendo investigaciones en este campo, como por ejemplo [24]. Aquí se propone el estudio de 10 géneros musicales: clásica, jazz, rock, blues, reggae, pop y *metal*. Se utilizan dos tipos de aprendizaje de máquina "clásicos": *Support Vector Machines* y *Random Forest* de los cuales se hablará mas adelante en la sección de *machine learning*, y como característica solo se emplea la transformada de Fourier de tiempo reducido. La transformada de Fourier de tiempo reducido es una versión de la transformada de Fourier clásica usada para determinar el contenido en frecuencia dado un intervalo de tiempo en una señal, se usa para evaluar el espectro entre dos instantes de tiempo y así poder evaluar los cambios en frecuencia a lo largo de la señal. En el caso de tiempo discreto la transformada de una señal $x[n]$ viene dada por la ecuación 2.1:

$$STFT\{x[n]\} \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]\, w[n-m]\, e^{-j\omega n} \tag{2.1}$$

Además se utilizan varios tipos de ventana debido que al usar ventanas cuadradas (las mas simples) se pueden obtener artefactos en frecuencia conocidos como *Fenómenos de Gibbs*. Las ventanas estudiadas se pueden ver en la figura 2.4, todas ellas se usarán para observar con cual se obtienen mejores resultados, así como también se estudiará como afecta el solape de las ventanas. Las características que finalmente se usan en [24] son la integración de los coeficientes obtenidos a lo largo del tiempo (que resulta en la distribución

de potencia en el espectro de frecuencias) y la desviación estándar de las frecuencias a lo largo del tiempo. Los resultados son bastante considerables si se tiene en cuenta que solo se hace uso de una característica y son 10 géneros musicales, siendo de un 71.3 % en el mejor de los casos para una *Support Vector Machine* con un *kernel* polinómico con una ventana *Parzen* de 512 muestras y 256 muestras de solape. La matriz de confusión se puede observar en la figura 2.5 extraída del artículo original.

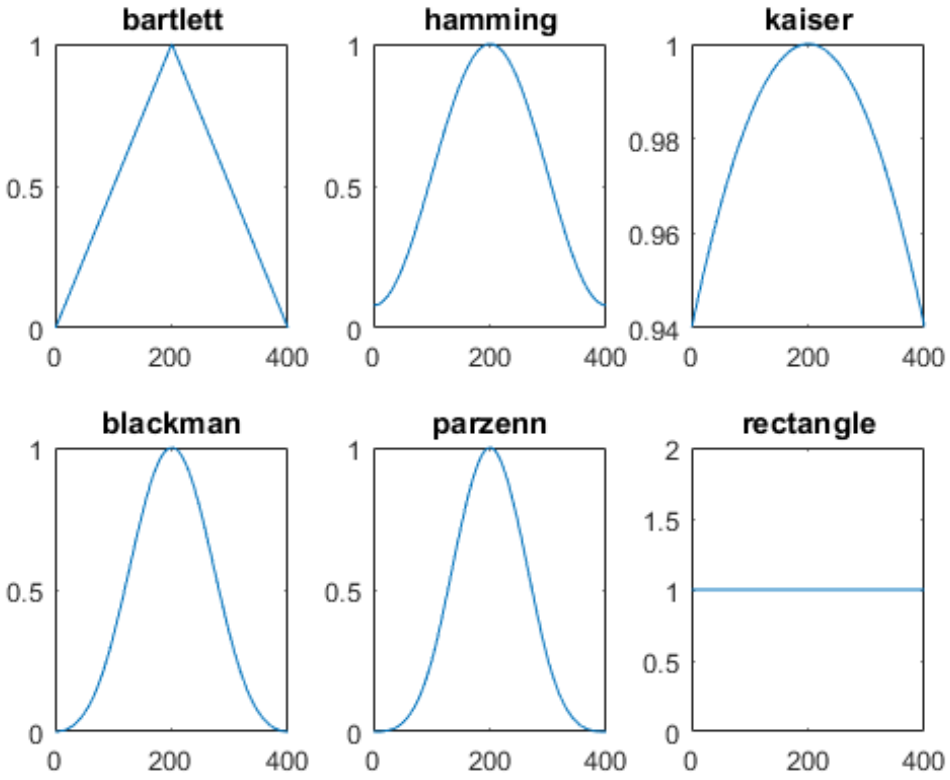

Figura 2.4 Ventanas usadas en el artículo [24].

Otro caso de aprendizaje de máquina clásico lo encontramos en el trabajo descrito en [25]. Se usan unas características seleccionadas en base a su popularidad en otros trabajos de [26][27][28][29][30][31], a las cuales añaden características como la clave en la que se encuentra la canción, la disonancia, o los histogramas del *beat* o ritmo. Su objetivo es clasificar siete géneros propios de la música brasileña: *forró, rock, repente, MPB, brega, sertanejo y disco*, utilizando las mencionadas características mediante máquinas de soporte vectorial. Los resultados alcanzados son de una media 79 % de aciertos a través de 30 iteraciones del proceso de validación. La matriz de confusión resultante se puede ver en la figura 2.6.

No obstante no solo son *support vector machines* y redes neuronales los objetos de estudio en este campo. En trabajos como [32] se comparan la clasificación de géneros musicales usando algoritmos del centroide mas cercano y de *k-nearest Neighbours*, los cuales describe el artículo. La selección de géneros en este caso son: baladas, blues, música clásica, armonía, *hip hop*, jazz, *keroncong*, latina, pop, electrónica, reggae y rock. Los

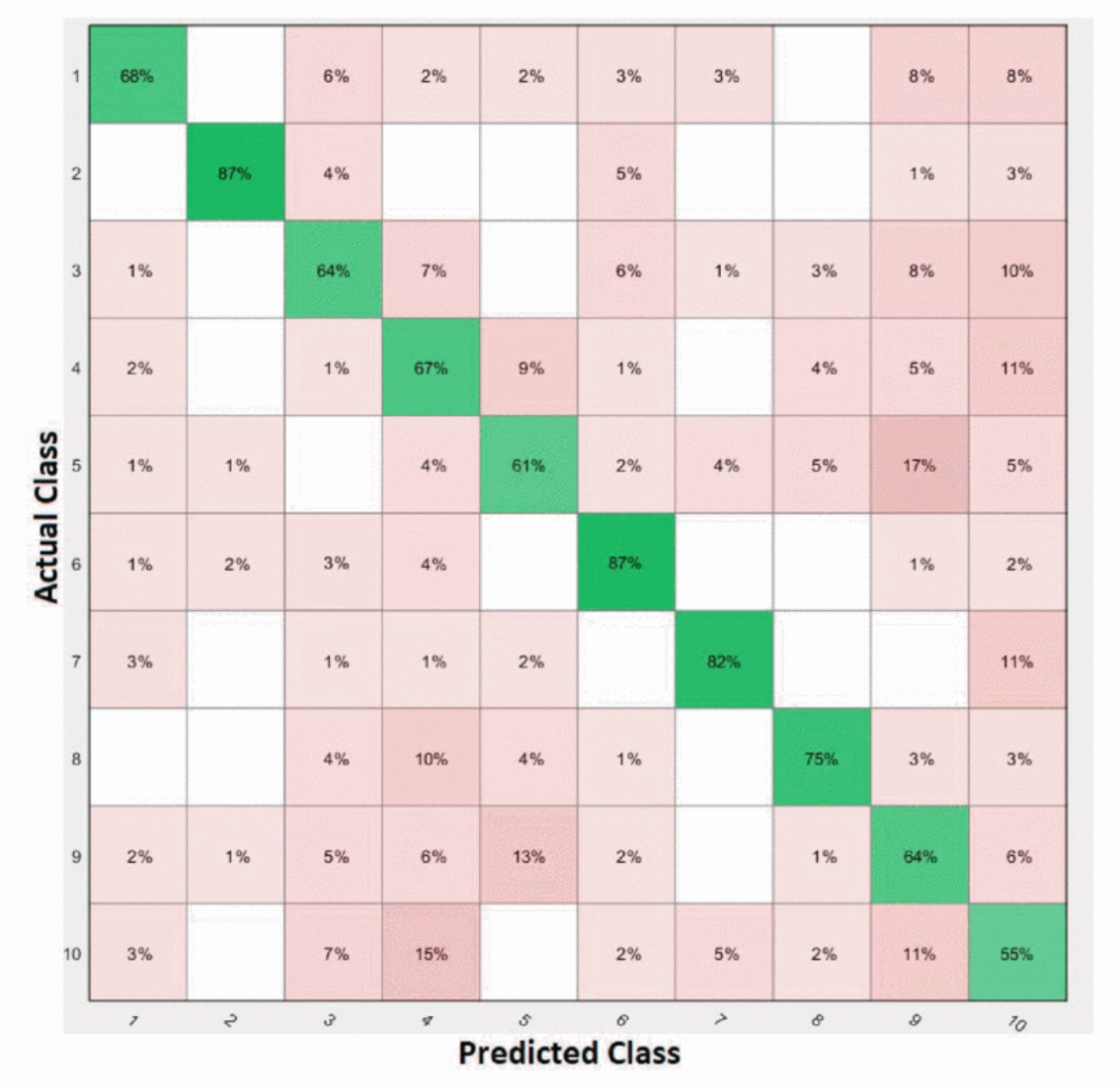

Figura 2.5 Matriz de confusión para el mejor caso de [24], extraída del artículo original.

Genres	Blues	Classical	Country	Disco	Hip Hop	Jazz	Metal	Pop	Reggae	Rock	Accuracies
Blues	863	7	31	10	6	25	38	5	4	45	83.46%
Classical	0	952	15	2	0	33	0	11	1	3	93.60%
Country	18	42	786	49	0	7	18	34	35	139	69.68%
Disco	12	30	35	639	24	2	13	46	97	72	65.87%
Hip Hop	7	1	0	62	805	0	3	30	44	6	84.02%
Jazz	19	9	12	10	0	852	0	9	6	14	91.51%
Metal	0	31	0	4	9	3	851	17	0	81	85.44%
Pop	0	4	7	40	39	2	0	758	31	46	81.76%
Reggae	6	4	23	58	54	4	6	20	710	57	75.37%
Rock	3	59	140	78	18	27	119	58	49	536	49.31%

Figura 2.6 Matriz de confusión del artículo [25] extraída del artículo original.

términos armonía y *keroncong* como géneros musicales hacen referencia en este caso a música instrumental y una forma de música nativo-indonesa respectivamente. Al igual que en [24] tenemos un espacio de características muy reducido, aunque en este caso no es tan extremo y son 3 en vez de 1. Las características aparecidas en este trabajo son: la tasa de cruces por cero, la energía media y la tasa de silencio. Las dos primeras se verán explicadas

mas adelante en el apartado de características, la tasa de silencio se define como:

$$SR = \frac{\sum s}{\sum l} \tag{2.2}$$

donde s son las muestras por debajo de cierto umbral y $\sum l$ es el número total de muestras. Es una forma de indicar la proporción de silencio existente en una pieza de sonido, definido el silencio como un rango para el cual el valor absoluto de la amplitud cae por debajo de cierto nivel. Los resultados en este trabajo muestran que estos clasificadores no son los más apropiados, ya que en el caso del centroide más cercano la tasa de aciertos es del 33,3 %, mientras que en el caso de *k-Nearest Neighbours* cae hasta el 22,5 % volviendo a poner a las máquinas de soporte vectorial en el centro de atención de la investigación actual. La tabla de confusión para el centroide mas cercano extraida del artículo original se puede ver en la figura 2.7.

Genre	A	B	C	D	E	F	G	H	I	J	K	L
A	-	-	1	-	-	-	-	1	-	-	-	-
B	-	-	-	-	-	-	-	-	-	-	1	-
C	-	-	10	-	-	-	-	-	-	-	-	-
D	-	-	-	-	-	-	-	-	-	-	1	1
E	-	-	-	-	-	1	-	-	-	1	-	-
F	-	-	-	-	-	-	2	-	-	-	-	-
G	-	-	-	-	-	-	-	-	1	1	-	-
H	1	-	-	-	-	-	1	-	-	-	-	-
I	-	-	-	-	-	-	-	2	-	-	-	-
J	-	-	-	-	-	-	1	-	-	-	-	-
K	1	-	-	-	-	-	-	-	-	-	-	-
L	-	-	-	-	-	-	-	1	-	-	1	-

Legend : A = Ballad; B = Blues; C = Classical; D = Harmony; E = Hip hop; F = Jazz; G = Keroncong; H = Latin; I = Pop; J = Electronic; K = Reggae; L = Rock

Figura 2.7 Matriz de confusión resultante del artículo [32]. Extraída del artículo original.

No solo el aprendizaje de máquinas es el único método explorado para el reconocimiento de géneros musicales también existen casos de clasificación mediante una *FPGA* y un procesador de señales digitales (DSP de ahora en adelante) como por ejemplo el de [33]. Una *FPGA* o matriz de puertas programables es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip. La característica que usan en este caso son los *Mel*

Frecuency Cepstral Coefficents (MFCC) de los que hablaremos mas adelante en la sección de características. Estos se extraen mediante la mencionada *FPGA*.

Sin embargo lo realmente interesante es la técnica que usan para clasificar, *Dynamic Time Warping* (DTW). DTW es una técnica usada para comparar patrones, la idea es ensanchar o comprimir dos secuencias temporales para ver cómo de parecidas son. La distancia entre ambas es calculada mediante la suma las distancias de los elementos alineados, cuanto menor sea la suma, más se parecerán dos señales. Para un determinado conjunto de canciones que podíamos llamar como "entrenamiento" se calculan los MFCC y se computa la media de los mismos, de tal manera que estos vectores serán comparados con las futuras canciones que lleguen al sistema para clasificación, asignándole en cada caso el vector medio de "entrenamiento" que produzca un menor valor de DTW. Sea $t(i), i = 1,2,\ldots,M$ la mejor señal y $s(j), j = 1,2,\ldots,N$ la señal de referencia. Se construye una matriz MxN tal mediante la ecuación 2.3.

$$D(i,j) = min\,[D(i-1,j-1), D(i-1,j), D(j,J-1)] + d(i,j) \tag{2.3}$$

donde $d(i,j)$ se puede calcular como:

$$d(i,j) = abs(t(i) - s(i)) \tag{2.4}$$

Donde las condiciones frontera son:

$$D(1,1) = d(1,1) \tag{2.5}$$

$$D(i,1) = \infty, i = 2,\ldots,M \tag{2.6}$$

$$D(i,j) = d(1,j), j = 2,\ldots,N \tag{2.7}$$

La distancia es el mínimo aparecido en la última columna de la matriz $D(i,j)$. Usando la base de datos [34] seleccionaron un subconjunto de los 10 existentes en la base, el subconjunto está formado por: blues, música clásica, jazz, *metal* y pop. De este subconjunto se seleccionaron 50 canciones por género de las cuales usaron 10 para clasificar. Con este sistema se obtuvo un 90 % de aciertos tal y como se observa en la matriz de confusión extraída del artículo en la figura 2.8.

Aunque este sistema en un principio parece no tener relación alguna con el aprendizaje de máquina, se puede entender como una aproximación mediante DTW al algoritmo de centroide más cercano. Las ventajas de este sistema son sin duda su rápida capacidad para el entrenamiento y la clasificación, puesto que no existen sistemas de cálculo "general" como podría ser un microprocesador, sino que es un circuito con sus conexiones realizadas para este mismo propśito.

Hasta este punto todas las características relevantes o bien eran desconocidas en el caso

Genre	Blues	Classical	Jazz	Metal	Pop
Blues	**8**	0	0	2	0
Classical	0	**8**	2	0	0
Jazz	0	1	**9**	0	0
Metal	0	0	0	**10**	0
Pop	0	0	0	0	**10**

Figura 2.8 Matriz de confusión para el sistema de [33].

de las redes neuronales, o bien eran de "bajo nivel" para el resto de técnicas (entendiendo como "bajo nivel" que solo actúa a niveles de frecuencia y/o tiempo, sin tener en cuenta la interpretación humana de las reglas para la música). Sin embargo en trabajos como [11] se usan características de alto nivel, siendo estas los acordes usados en cada canción.

En [11] los acordes se obtienen a través de la página web *e-chords.com*. De ahí un algoritmo extrae los acordes de las canciones en base al fichero *html*, en concreto se obtuvieron 8994 canciones de los géneros: *axé, bossa nova, forró, rap, sertanejo y samba*. A continuación en base al nombre del artista y título de la canción, otro programa descargaba los 60 segundos siguientes a los 30 primeros iniciales para cada canción obtenida de la web, de esa forma se tenía por un lado representando a la canción el propio audio y por otro lado un fichero con la secuencia de acordes. Para simplificar los posibles resultados, todos los acordes se acababan reduciendo a sus tríadas, es decir su expresión mas básica, entonces si se tenía una secuencia tal que "D/F# Fm5-/7", se transforma a "D Fm".

Cada acorde posible supone una nueva dimensión en las que los valores solo pueden ser o que existe el acorde en esa canción o que no. Como soporte a esta característica se añadieron otras como patrones de ritmo, descriptores estadísticos del espectro y histograma de ritmo. Usando solo la característica de acordes se obtiene una tasa de aciertos del 52,44 %, una vez realizadas las pertinentes simplificaciones de acordes y con un clasificador del tipo *random forest* del cual hablaremos más adelante en el capítulo de aprendizaje de máquinas. Sin embargo el máximo se alcanza cuando solo se añaden los descriptores estadísticos del espectro a los acordes, llegando a un 78.4 % de aciertos.

Sin embargo todas estas técnicas tienen un problema, y es que necesitan un gran volumen de datos para entrenar y probar los algoritmos.

Existen gran variedad de bases de datos en internet, como por ejemplo la *latin music database* [35]. Esta incluye 10 estilos de música distintos, no obstante no contiene las canciones al completo, por cuestiones de copyright, sino que lo que ofrece esta base de datos son las características extraídas de estas canciones al principio, mitad y final de cada

canción, acorde a la extracción explicada en [36]. Es por el deseo de probar con nuestras propias características y tratar directamente el audio por el que nos decantamos a no usar esta base de datos finalmente. Existen otros ejemplos de bases de datos colaborativas, como la promovida por la *Universidad Pompeu Fabra* [37] que posee incluso una API en la que poder obtener o subir *datasets* completamente. Sin embargo posee limitaciones similares a la anterior debido al Copyright. Dadas las circunstancias en las que nos situamos con respecto a las bases de datos, la decisión que tomamos fue la de crear nuestra propia base de datos y etiquetarla manualmente.

3 Aprendizaje de máquinas

"Ah, music," he said, wiping his eyes. "A magic beyond all we do here!

J.K. Rowling, Harry Potter and the Sorcerer's Stone

Machine learning el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que los ordenadores *aprendan*. Tiene una amplia gama de aplicaciones como motores de búsqueda, diagnósticos médicos, análisis de mercado de valores, reconocimiento de patrones, etc…y en este caso lo usaremos para clasificar los distintos géneros. Además se explicará mas detalladamente las técnicas y algoritmos que han dado lugar a los mejores resultados.

3.1 Tipos de algoritmos

A grandes rasgos las técnicas de aprendizaje de máquina se clasifican en dos grandes familias: aprendizaje supervisado y aprendizaje no supervisado.

3.1.1 Aprendizaje supervisado

El algoritmo produce una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema. Para ello se requiere de un de entrenamiento que contenga varios vectores de características del sistema y las clases a la que pertenecen, por lo que un claro uso de este tipo de algoritmo es el de clasificación de objetos, imagenes, secuencias de ADN, etc. Dentro de esta categoría, aparte de los algoritmos de clasificación existen tambien los denominados de *regresión* los cuales se usan para predecir respuestas continuas, para por ejemplo predecir el consumo eléctrico, etc. Existen multitud de algoritmos de esta clase.

En el caso de *Support Vector Machines* o Máquinas de soporte vectorial las clases son separadas mediante un algoritmo que construye un hiperplano o un conjunto de hiperplanos [38], con un vector de soporte definido como el vector entre dos puntos de las clases, tales que el hiperplano resultante genera el mayor margen posible para ambas clases como se ve en la figura 3.1.

Existen además otros algoritmos importantes como discriminantes o los *ensembles*, que veremos con mas detalle en las siguientes secciones.

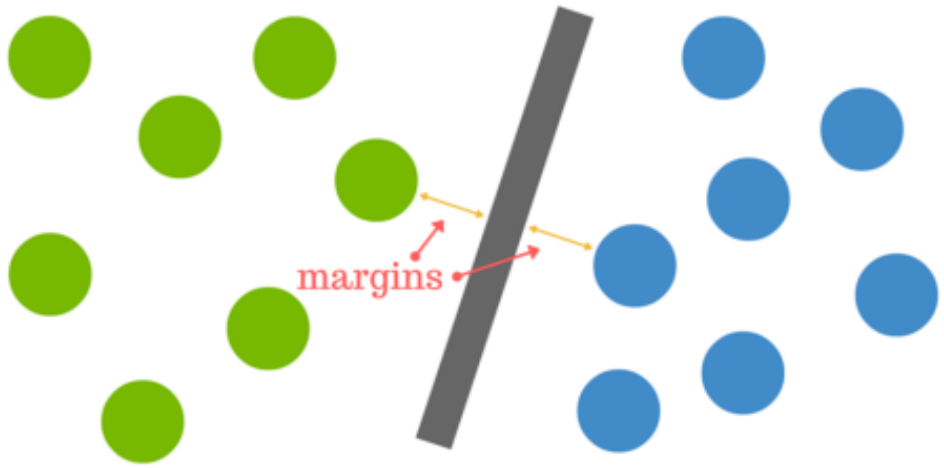

Figura 3.1 Separación de dos clases mediante un hiperplano guardando el mayor margen posible [39].

3.1.2 Aprendizaje no supervisado

Este tipo de algoritmos se usa para encontrar estructuras y patrones de un set de datos. A diferencia del caso anterior de *Aprendizaje supervisado* en este caso solo se tiene una serie de vectores de características como entradas al sistema, sin ninguna clase de etiquetas. Encuentra su utilidad en muchos algoritmos de compresión de datos. La mayoría de algoritmos de este tipo son de *clustering* (que explicaremos con mas detalle en una sección posterior), pudiendo ser estos *Hard-clustering* o *Soft-clustering*, en el primer caso cada punto de datos puede pertenecer a un *cluster* solo, mientras que en el segundo caso puede pertenecer a varios *clusters*.

3.2 Ensembles

El problema estándar de aprendizaje supervisado puede entenderse como que dados m vectores de características de entrenamiento de la forma $(x_1, y_1), \ldots, (x_m, y_m)$, se busca alguna función tal que $y = f(x)$, siendo x_i vectores de la forma $\left\langle x_{i,1}, x_{i,2,\ldots,x_{i,n}} \right\rangle$ cuyos componentes son las n características asociadas a cada objeto de la clase y_i, e y_i los valores

(en este caso discretos) conocidos como las etiquetas $\{1,2,\ldots,K\}$ que hacen alusión a las posibles clasificaciones del objeto. Dado un set de muestras de entrenamiento el algoritmo de aprendizaje devuelve un clasificador, que es una hipótesis acerca de la verdadera función f.

Un *ensemble* es un set de clasificadores cuyas decisiones individuales son combinadas (con distintos o equitativos pesos a la hora de votar) para clasificar nuevas muestras. La idea es que los *ensembles* son normalmente mucho mas precisos que los clasificadores que lo componen por separado.

Sin embargo, para que esto funcione así ha de cumplirse dos requisitos [40]:

- Que los clasificadores tengan una tasa de mayor de error que una elección aleatoria sobre los posibles resultados.
- Que sean diversos. Esto quiere decir que cuando los clasificadores cometen un error de clasificación de un nuevo objeto, los resultados erróneos que dan son distintos.

Esta segunda característica es muy importante [41], supongamos que tenemos tres clasificadores h_1, h_2, h_3 y un nuevo objeto x. Si los tres son idénticos cuando h_1 se equivoque, h_2 y h_3 también lo harán. En cambio si los errores son incorrelados cuando h_1 se equivoque h_2 y/o h_3 podrían ser correctos. Para un número de L componentes de un *ensemble*, podemos estimar aproximadamente la posibilidad de que más de L/2 hipótesis estén equivocadas, suponiendo por ejemplo que la probabilidad de error es la misma para todas las hipótesis y usando una variable binomial donde L es el número de ensayos, y la probabilidad de error de cada hipótesis es la probabilidad de "éxito" de la binomial, obteniendo así la figura 3.2, que además mantiene una forma similar siempre y cuando $P_e < 0.5$

Si las hipótesis tienen una tasa de error mayor que 0.5 la forma de la gráfica será muy distinta (véase figura 3.3), y por lo tanto deberá o optimizar el espacio de las características, buscar otras características, o directamente probar con otro algoritmo de *machine learning*.

3.2.1 Construcción de *Ensembles*

Existen diversos algoritmos para crear *ensembles*:

- Clasificador óptimo Bayesiano
- Manipulación de las muestras de entrenamiento:
  - *Bagging* [42]
  - Subespacios Aleatorios (*Random Subspaces*)
  - *AdaBoost* [43]
- Manipulación de las características de entrada [44][45]
- Manipulación de las clases de salida [46]

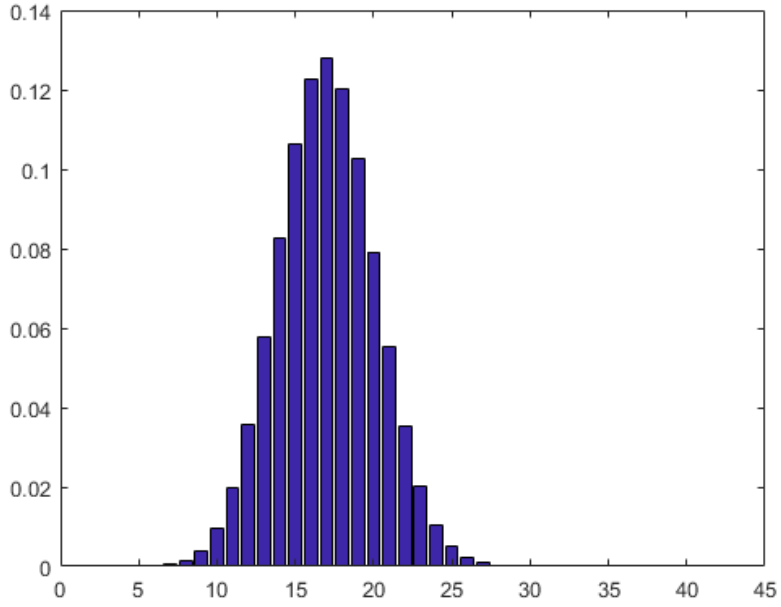

Figura 3.2 La probabilidad de que l de 40 hipótesis sean incorrectas, siendo errores incorrelados y con una $P_e = 0.4$ La probabilidad de que más de la mitad de los clasificadores hayan cometido un error, será de 0.1298, lo cual es bastante menor que 0.4.

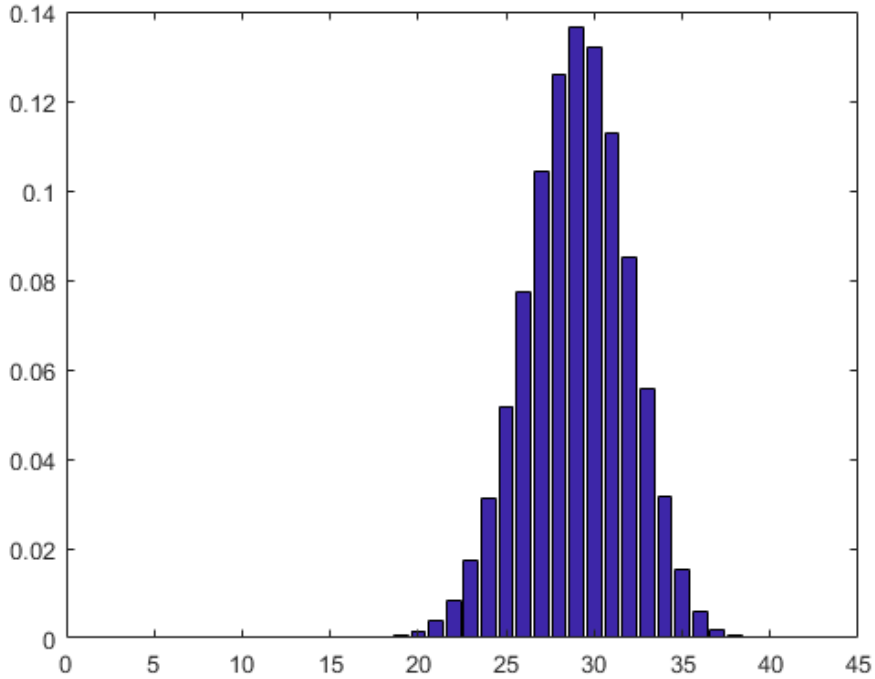

Figura 3.3 La probabilidad de que l de 40 hipótesis sean incorrectas, siendo errores incorrelados y con una $P_e = 0.7$. La probabilidad de que más de la mitad de los clasificadores hayan cometido un error, será de 0.9976, lo cual es bastante mayor que 0.7.

Cada uno de estos métodos tiene sus ventajas y sus desventajas. El método de *bagging* por

ejemplo genera una serie de modelos a partir de un subconjunto del conjunto de muestras originales con una cantidad de N reemplazos al subconjunto elegido. Cada uno de los modelos tiene asignado un subconjunto como muestras de entrenamiento, de forma que cada modelo se entrena por separado. Una vez terminado el proceso de entrenamiento el ensemble resultante usará estos modelos contando por igual todos sus votos, entendiendo por voto la clase a la que estiman que pertenece una determinada muestra.

AdaBoost es por su parte un algoritmo basado en *Boosting* que al igual que *Bagging* se basa en usar los votos de varios modelos, aunque en este caso los modelos tienen ciertos valores de peso en función a su precisión. A cada iteración los clasificadores que antes tuvieron muchos errores prediciendo ciertas muestras del set de entrenamiento sufren mayor penalización, de esta forma se fuerza a que los peores clasificadores sean reentrenados y mejorados con mas prioridad para obtener mejor resultados. Funciona mejor cuando los clasificadores son mas sencillos, puesto que al contrario que *Bagging* o [45] (que explicaremos mas adelante) los clasificadores de los que se compone el *ensemble* no se entrenan en paralelo, sino que han de hacerlo conforme se penalice a los peores clasificadores.

Por norma general el clasificador óptimo bayesiano es el que mejor funciona con diferencia [47].

3.2.2 Método de *Random Subspaces*

El método de subespacios aleatorios está descrito en [45]. En este método se basa en un procedimiento autónomo y pseudoaleatorio para seleccionar un número pequeño de dimensiones dado el espacio completo de características de un sistema. En cada iteración se escoge un subespacio de N dimensiones del espacio completo, todos los valores de este subespacio se fijan a un valor (sea 0 por ejemplo) y a continuación todas las características del espacio completo, son proyectadas sobre el subconjunto de esa iteración (véase figura 3.4) , siendo finalmente este el conjunto que se utilizará para entrenar a uno de los clasificadores del ensemble. Cuando se vaya a probar con la clasificación de una muestra desconocida, el vector de características se proyecta sobre el subespacio correspondiente y se usa el correspondiente clasificador.

Para un espacio de características de n dimensiones, hay hasta 2^n elecciones de subconjuntos posibles por hacer, pudiendo ser finalmente los subespacios que tomemos de igual tamaño para todos los clasificadores o de distinto tamaño para cada uno, lo cual es una buena forma de explorar todos los resultados posibles (aunque también hace crecer exponencialmente el tiempo de cálculo del ensemble óptimo).

Otra de las ventajas de este método es que reduce el impacto de *la maldición de la dimensionalidad*. Este fenómeno ocurre cuando los vectores de características son muy altos, lo que a algoritmos de aprendizaje automático basados en el gradiente le supone un gran problema por manejar gradientes de muchas componentes. También sufren de esta maldición algoritmos basados en el vecino mas cercano, puesto que conforme se añaden nuevas características aumenta la distancia entre vecinos.

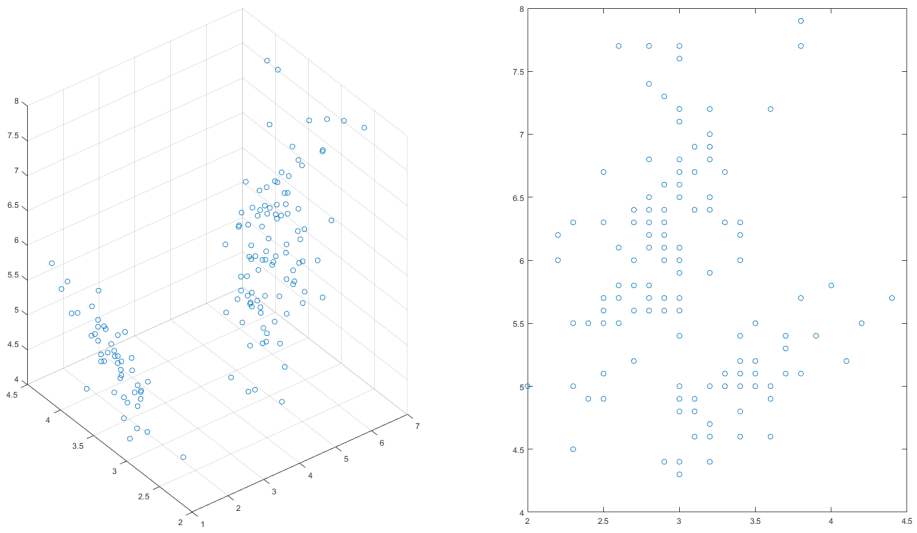

Figura 3.4 Espacio de características de 3 dimensiones y su proyección en un subconjunto aleatorio de 2 dimensiones.

La proyección en subespacios de esos vectores de alta cantidad de características, puede reducir la carga computacional para llegar al mínimo del gradiente en caso de algoritmos basados en el gradiente de la función de coste, o en el caso de los vecinos mas cercanos evitar que crezca de manera exponencial las distancias entre los mismos.

Otra de las ideas previamente exploradas para crear múltiples clasificadores proviene de la idea del método de *cross-validation*. En este caso se seleccionan subconjuntos aleatorios (no de características, sino de muestras) y cada clasificador se entrena con ese subconjunto de muestras *(bagging)*.

En el método original propuesto en [45] las combinaciones de los distintos clasificadores se dan de la siguiente forma (recordando que en este artículo aplica la idea de subespacios aleatorios a árboles de decisión).

Las decisiones de los n_t árboles son combinadas promediando la posibilidad de cada clase en las hojas del árbol. Para un punto x, sea $v_j(x)$ la hoja donde termina el punto x cuando desciende por el árbol T_j (j = 1, 2, 3, 4, ..., n_t). Dicho esto, denotamos la posibilidad de que el punto x pertenezca a la clase c (c = 1, 2,..., n_c) por $P(c \,|v_j(x))$ y puede ser estimada de la siguiente forma:

$$\hat{P}(c|v_j(x)) = \frac{P(c \,|v_j(x))}{\sum_{k=1}^{n_c} P(c_k, v_j(x))} \tag{3.1}$$

Es decir, la fracción de puntos de c entre todos los puntos asignados a $v_j(x)$ (en el set de

entrenamiento). No obstante la mayoría de hojas solo contiene una clase así que los valores de la estimación $\hat{P}(c \,|\, v_j(x))$ suelen ser siempre 1. La función discriminante se define como:

$$g_c(x) = \frac{1}{n_t} \sum_{k=1}^{n_t} P(c \,|\, v_j(x)) \quad (3.2)$$

y la regla de decisión es asignar x a la clase c cuyo $g_c(x)$ sea máximo. Para árboles en cuyos nodos no haya compartido ninguna clase en ninguna de las hojas, esta regla es equivalente a elegir la clase que sume mas votos entre todos los árboles. Mientras no haya ambiguedad en los subespacios elegidos el discriminante preservará la precisión del 100 % en el set de entrenamiento. Sin embargo es posible que dos muestras puedan ser indistinguibles la una de la otra en un determinado subespacio. Por lo tanto hay que tener mucho cuidado al aplicar este método a sistemas cuya dimensionalidad es baja, puesto que esta ambiguedad podría repetirse varias veces en varios subespacios, dando lugar a clasificadores del *ensemble* poco precisos, y en consecuencia teniendo un *ensemble* peor, como se se mostraba en la figura 3.3.

3.3 Análisis mediante Discriminantes

El análisis discriminante se puede considerar como un análisis de regresión donde la variable dependiente es categórica y tiene como categorías las etiquetas de cada uno de los grupos, y las variables independientes son continuas y determinan a que grupo pertenecen los objetos. Se pretende encontrar relaciones lineales entre las variables continuas que mejor discriminen en los grupos dados a los objetos, con la idea de construir una regla de decisión que asigne un objeto nuevo. Un dato importante para este caso es que el número de variables discriminantes debe ser menor que $m-2$, siendo m el número de objetos y ninguna variable discriminante puede ser combinación lineal de otras variables discriminantes. Así mismo tendremos $n-1$ funciones discriminantes, donde n es el número de clases disponibles.

Sea el problema clásico de aprendizaje supervisado:

$$y_1 = a_{11}x_1 + \ldots + a_{1p}x_p + a_{10} \quad (3.3)$$

$$y_m = a_{m1}x_1 + \ldots + a_{mp}x_p + a_{m0} \quad (3.4)$$

Procederemos a calcular la variabilidad dentro de los grupos y entre los grupos. Partiendo de la definición de covarianza:

$$Cov(x_j, x_{j'}) = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \bar{x_j})(x_{ij'} - \bar{x_{j'}}) \quad (3.5)$$

Siendo $\bar{x_j}$ la media total de la variable x_j que se puede calcular como:

$$\bar{x_j} = \frac{1}{n}\sum_{i=1}^{n} x_{ij} = \frac{1}{n}\sum_{k=1}^{q}\sum_{i\in I_k} x_{ij} = \frac{1}{n}\sum_{i=1}^{q} n_k \bar{x_{ij}} \tag{3.6}$$

Donde q es el número de clases e I_k el grupo de la clase k-ésima clase. Seguimos el procedimiento en [48]:

$$\bar{x_{ij}} = \frac{1}{n_k}\sum_{i\in I_k} x_{ij} \tag{3.7}$$

Ahora podemos calcular la covarianza como:

$$Cov(x_j,x_{j'}) = \frac{1}{n}\sum_{k=1}^{q}\sum_{i\in I_k} (x_{ij} - \bar{x_j})(x_{ij'} - \bar{x_{j'}}) \tag{3.8}$$

si sustituimos:

$$(x_{ij} - \bar{x_j}) = (x_{ij} - \bar{x_{kj}}) + (x_{kj} - \bar{x_j}) \tag{3.9}$$

y

$$(x_{ij'} - \bar{x_{j'}}) = (x_{ij'} - \bar{x_{kj'}}) + (x_{kj'} - \bar{x_{j'}}) \tag{3.10}$$

en 3.8 y simplificamos, obtenemos:

$$Cov(x_j,x_{j'}) = \frac{1}{n}\sum_{k=1}^{q}\sum_{i\in I_k} (x_{ij} - \bar{x_k}j)(x_{ij'} - \bar{x_{kj'}}) + \sum_{k=1}^{q}\frac{n_k}{n}(\bar{x_{kj}} - \bar{x_j})(\bar{x_{kj'}} - \bar{x_{j'}}) = d(x_j,x_{j'}) + e(x_j,x_{j'}) \tag{3.11}$$

Es decir, la covarianza "total" es igual a la covarianza "dentro" de grupos mas la covarianza "entre" grupos. Definimos pues la covarianza total, por cuestiones de legibilidad como:

$$t(x_j,x_{j'}) = d(x_j,x_{j'}) + e(x_j,x_{j'}) \tag{3.12}$$

que escribiremos en notación matricial de ahora en adelante como:

$$T = E + D \tag{3.13}$$

donde:

- T = matriz de covarianzas total
- E = matriz de covarianzas entre grupos
- D = matriz de covarianzas dentro de grupos

3.3.1 Extracción de las funciones discriminantes

La idea básica del Análisis Discriminante consiste en extraer a partir de $x_1,\ldots,x_p$ variables observadas en k grupos, m funciones $y_1,\ldots,y_m$ de forma:

$$y_i = a_{i1}x_1 + \ldots + a_{ip}x_p + a_{i0} \tag{3.14}$$

donde m = min(q-1,p), tales que $corr(y_i,y_j) = 0$ para todo $i \neq j$.

Si las variables $x_1,\ldots,x_p$ están tipificadas, entonces las funciones:

$$y_i = a_{i1}x_1 + \ldots + a_{ip}x_p \tag{3.15}$$

para $i = 1,\ldots,m$ se denominan funciones discriminantes canónicas. Las y_m funciones se extraen de modo que:

1. y_1 sea la combinación lineal de $x_1,\ldots,x_p$ que proporciona la mayor discriminación posible entre los grupos.

2. y_2 sea la combinación lineal de $x_1,\ldots,x_p$ que proporciona la mayor discriminación posible entre los grupos, después de y_1, tal que $corr(y_i,y_2) = 0$.

En general, y_i es la combinación lineal de $x_1,\ldots,x_p$ que proporciona la mayor discriminación posible entre los grupos después de y_{i-1} tal que $corr(y_i,y_j) = 0$ para $j = 1,\ldots,(i-1)$. Un ejemplo de función discriminante producida puede verse en 3.5.

3.3.2 Procedimiento matricial

La mayoría del hardware actual se ha centrado en aumentar la capacidad de cómputo paralelo. Las CPU se han dejado de medir solo por la velocidad a la que son capaces de realizar operaciones, ahora también se tiene en cuenta la capacidad de las mismas para el cálculo paralelo, cuantos más núcleos, más operaciones en paralelo podremos hacer y mas tiempo podremos ahorrar. No solo eso, si no que sectores tradicionales como el de hardware para videojuegos y ocio han visto una oportunidad en este campo y han desarrollado arquitecturas bajo este principio de paralelismo, como ejemplo tenemos el caso de NVIDIA y CUDA [17] que además de obtener un mayor rendimiento en el procesado de imagen también resulta especialmente útil en el caso de aprendizaje automático.

Todo esto lleva a una nueva necesidad de aprovechar todo el potencial que ofrece este nuevo hardware, es por eso que un lugar de avance interesante es la *vectorización* o *paralelización* de este. Por lo que vamos a proceder al desarrollo del mismo según aparece en [48].

Buscamos una función lineal de $x_1, \ldots, x_p : y = a'x$, de modo que

$$Var(y) = a'Ta = a'Ea + a'Da \tag{3.16}$$

que como senñalamos al principio de la sección es la variabildad entre grupos mas la variabilidad dentro de grupos.

Queremos maximizar la variabilidad entre los grupos para discriminarlos mejor. Para conseguirlo debemos maximizar

$$\frac{a'Ea}{a'Ta} \tag{3.17}$$

Es decir, maximizar la varianza entre grupos en relación al total de la varianza. Considerando la función:

$$f(a) = \frac{a'Ea}{a'Ta} \tag{3.18}$$

f es una función homogénea puesto que

$$f(a) = f(\mu a) \forall \mu \in \Re \tag{3.19}$$

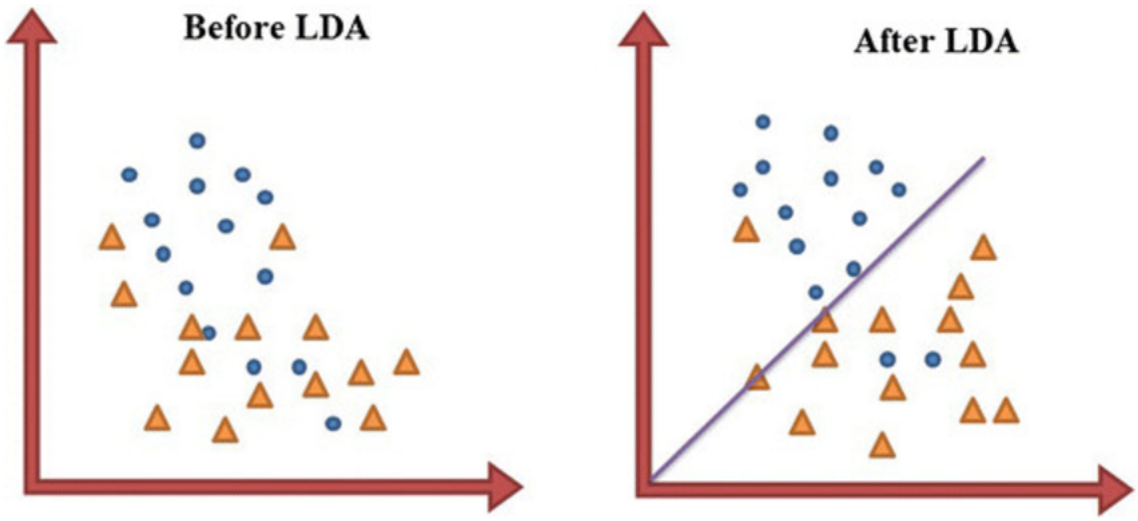

Figura 3.5 Función producida por algoritmo de análisis de discriminantes lineales [49].

por lo que solo es necesario maximizar $a'Ea$ tal que $a'Ta = 1$. Este es un problema que se puede resolver utilizando multiplicadores de *Lagrange*. Sea:

$$L = a'Ea - \lambda(a'Ta - 1) \tag{3.20}$$

y su derivada:

$$\frac{\partial L}{\partial a} = 0 \tag{3.21}$$

$$\frac{\partial L}{\partial a} = 2Ea - 2\lambda Ta = 0 \tag{3.22}$$

$$Ea = \lambda Ta \Rightarrow (T^{-1}E)a = \lambda a \tag{3.23}$$

Por tanto, el autovector asociado a la primera función discriminante lo es de la matriz $T^{-1}E$ y como $Ea = \lambda Ta$ entonces $a'Ea = \lambda a'Ta = \lambda$.

Luego si tomo el vector asociado al máximo autovalor, se obtendrá la función que recoge el máximo poder discriminante. A continuación se procede calculando los autovectores de la matriz $T^{-1}E$ asociados a los autovalores elegidos en orden decreciente:

$$a_2'x \;\Rightarrow\; a_2'x = y_2 \tag{3.24}$$

$$a_m'x \;\Rightarrow\; a_m'x = y_m \tag{3.25}$$

donde m = min(q-1,p) Estos son vectores linealmente independientes y por lo tanto dan lugar a funciones incorreladas.

La suma de todos los autovalores es la proporción de varianza total que queda al considerar solo los ejes o funciones discriminantes. Por lo tanto el porcentaje restante de varianza de y_i viene dada por:

$$\frac{\lambda_i}{\sum_{i=1}^{m}\lambda_i} * 100\,\% \tag{3.26}$$

3.3.3 Clasificación

Una vez se tienen calculadas las funciones discriminantes se puede proceder al proceso de clasificación de otros objetos, para ello se le asigna un valor o "puntuación" a cada nuevo objeto utilizando su vector de características y las funciones discriminantes calculadas previamente. De esta forma, sean las reglas de decisión y_k y sus coeficientes en la forma a_{kj}, es decir la componente j-ésima de la función discriminante k-ésima, entonces el vector

de características x_i obtiene un valor/puntuación en la k-ésima función discriminante dado por:

$$y_{ik} = a_{k1}x_{i1} + \cdots + a_{kp}x_{ip} + a_{k0} \tag{3.27}$$

Estas puntuaciones pueden usarse de varias formas para clasificar la muestra x_i [50] por ejemplo siguiendo el procedimiento de [48], aquí se usa la llamada *regla de Bayes*. En esta, la posibilidad de que un objeto j, con una puntuación discriminante $D = (y_{j1}, \ldots, y_{jm})$, pertenezca al grupo i-ésimo se puede estimar mediante la conocida regla de Bayes:

$$P(G_i|D) = \frac{P(D|G_i)P(G_i)}{\sum_{i=1}^{k} D|G_iP(G_i)} \tag{3.28}$$

$P(G_i)$ es la probabilidad a priori, se puede interpretar como la estima de la confianza de que un objeto pertenezca a un grupo si no se tiene información previa.

$P(D|G_i)$ es la probabilidad de obtener la puntuación D estando en el grupo i-ésimo. Como las puntuaciones discriminantes se calculan a partir de combinaciones lineales de p variables, distribuidas según una normal, se distribuyen a su vez como una normal, cuya media y varianza se estiman a partir de todas las puntuaciones que se recogen en el grupo i-ésimo.

$P(G_i|D)$ es la probabilidad a posteriori que se estima a través de $P(G_i)$ y de $P(D|G_i)$. En realidad mide lo mismo que $P(G_i)$, pero refina la medida de incertidumbre al tener en cuenta la información que recogen las puntuaciones discriminantes D. Es decir, lo que interesa es calcular la probabilidad de que un objeto pertenezca al grupo G_i, dado que presenta la puntuación D.

Se asignará un objeto al grupo G_i, cuya probabilidad a posteriori sea máxima, es decir, dado que presenta la puntuación D.

Otro método existente es por ejemplo el que usa Matlab [51]. En este se asume que cada clase sigue una distribución Gaussiana multivariable según Fisher [52] con una media arbitraria y una matriz de covarianza común. Esto quiere decir que suponemos que existe una serie de $\mathcal{N}_1 \ldots \mathcal{N}_k$ tales que $(\mu_1,\Sigma_1), \ldots (\mu_k,\Sigma_k)$ y asumimos que $\Sigma_k = \Sigma_{XX}\, \forall\, k$ siendo Σ_{XX} la matriz obtenida de .

Matlab genera una función de coste con la cual se busca una y (que denominaremos por $\hat{y}$) que minimice:

$$\sum_{k=1}^{K} \hat{P}(k|x)(C(y|k)) \tag{3.29}$$

siendo $\hat{y}$ la clase a la cual predice que pertenece, k es el número de clases, $\hat{P}(k|x)$ la probabilidad a posteriori y $C(y|k)$ el coste asociado a clasificar y suponiendo que la clase verdadera sea k.

La función de coste $C(y|k)$ es muy sencilla, solo tiene dos valores posibles:

- 0 si $y = k$
- 1 si $y \neq k$

La función $\hat{P}(k|x)$ (probabilidad a posteriori) de que un punto x pertenezca a la clase k puede verse como el producto de la probabilidad a priori y la función de densidad de probabilidad multivariable, que hemos construido bajo la mencionada asunción en [52]. Esta función de densidad con media μ_k y covarianza Σ_k en el punto x es:

$$P(x|k) = \frac{1}{2\pi|\Sigma_k|} exp(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^- 1(x-\mu_k)) \tag{3.30}$$

donde $|\Sigma_k|$ es el determinante de $\Sigma_k y \Sigma_k^{-1}$ es la matriz inversa. Siendo pues $P(k)$ la probabilidad a priori de una clase k, entonces la probabilidad a posteriori de que una clase x sea k es:

$$\hat{P}(k|x) = \frac{P(x|k)P(k)}{P(x)} \tag{3.31}$$

Aunque normalmente se supone que $P(k) = cte \, \forall \, k$.

3.4 K-means clustering

Los algoritmos de *clustering* permiten obtener relaciones de una gran cantidad de datos que en principio no tenían relación explícita agrupados en una forma similar a la de la figura 3.6. Es por ello que gozan de una gran aceptación en ámbitos como el estudio de transporte [53], recomendadores de contenido [54] o incluso estudios sociológicos [55].

El algoritmo de *k-means* es uno de los mas antiguos [56]. En resumen este algoritmo trata de encontrar *K clusters* que no se solapen entre sí. Cada *cluster* se encuentra representado mediante su centroide que es la media de los vectores de características de los objetos pertenecenientes a esa clase. Podemos definir el proceso de creación de los *clusters* como una serie de pasos que se van alternando tal y como viene en [57], a excepción del «Paso 0» que solo se hace una vez al inicio del todo:

- Paso 0. Crea K centroides aleatorios en el espacio de características.
- Paso 1. Crear los *clusters*: El algoritmo itera a lo largo de todo el set de datos y asigna cada objeto a un cluster representado por el centroide mas cercano. El centroide mas cercano vendrá dado según una medida de *disparidad*, como por ejemplo la distancia euclídea, sea cual sea la representamos por d. De esta forma para cada objeto en el *dataset* se busca minimizar d^2 tal que:

$$min_j \, d^2(x_i, c_j) \tag{3.32}$$

donde d es la función de disparidad, x_i es el i-ésimo objeto a clasificar, y c_j es el centroide del j-ésimo *cluster*

- Paso 2. Buscar los centroides de los *clusters*: Para cada *cluster* se busca minimizar la siguiente función objetivo:

$$e(X_j) = \sum_{i=1}^{n_j} d^2(x_i, c_j) \tag{3.33}$$

donde X_j es el j-ésimo *cluster*, x_i es el i-ésimo objeto clasificado en el j-ésimo *cluster*, c_j es el centroide del j-ésimo *cluster* y d es la función de disparidad. En el caso de que se quiera utilizar el "método de Ward" [58], el centroide se calculará como la media de todos los puntos pertenecientes al *cluster*.

Este proceso se puede repetir o bien un numero máximo de iteraciones o bien hasta que los centroides *clusters* varíen menos de un valor determinado. Uno de los problemas de los que sufre este algoritmo es el de converger a un mínimo local en lugar del global, y hay mucha investigación para solventar estos problemas de *k-means*, como por ejemplo [59] donde se muestra que la técnica de *bisecting k-means* mejora de forma considerable estos inconvenientes del algoritmo. Además la medida de disparidad también puede ser de objeto estudio y experimentación para usar medidas como la distancia Kullback-Leibler [60] o la divergencia de Bregman [61].

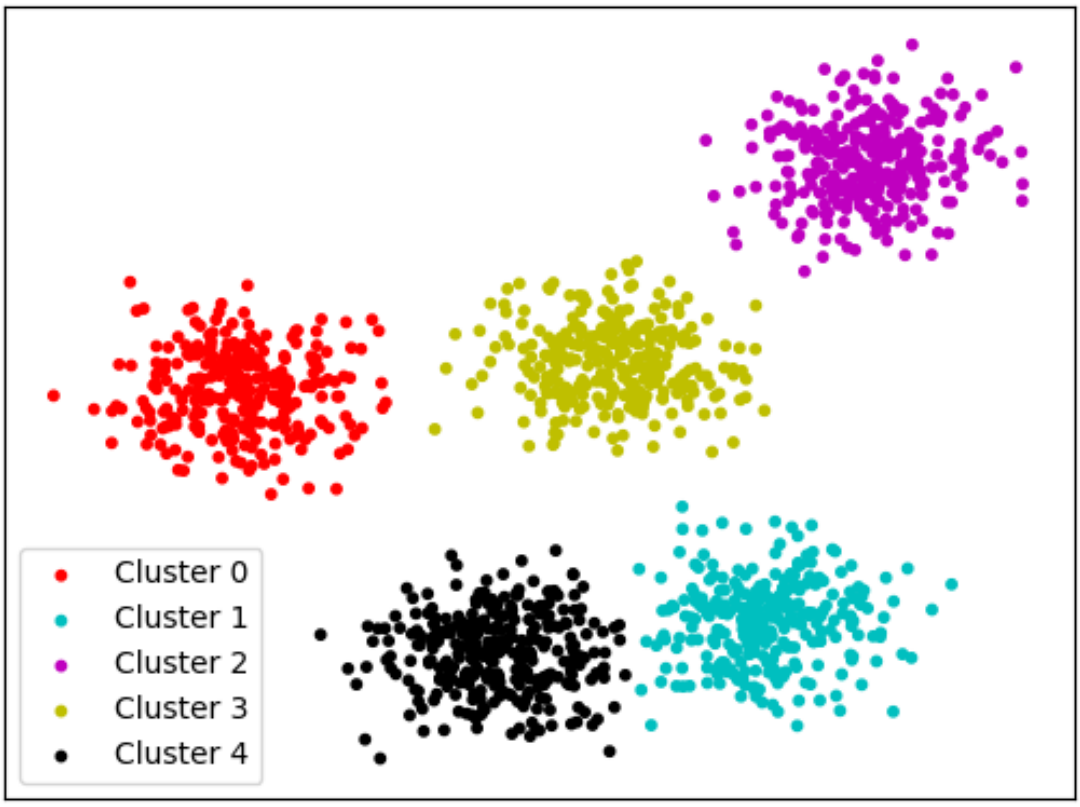

Figura 3.6 Clusters definidos muy claramente [62].

4 Características usadas

Hay dos formas de refugiarse de las miserias de la vida: la música y los gatos.

ALBERT SCHWEITZER

Una buena base para que el algoritmo de aprendizaje automático funcione es que tener una buena selección de características que permitan diferenciar correctamente a todas las clases. En este capítulo se explicarán las características usadas y los procedimientos utilizados.

Nota: Cada vez que se diga que se obtiene la varianza y la media a través de un enventanado, el tamaño de la ventana será de 2048 muestras con 512 muestras de solape y el número de puntos de la DFT es de 2048.

4.1 Tímbricas y espectrales

4.1.1 MFCC

Los MFCC *(Mel Frequency Cepstral Coefficients)* son coeficientes originalmente destinados a la representación del habla basados en la percepción auditiva humana. Esto es muy importante puesto que el cerebro humano no oye de forma normal si no que tiene mayor sensibilidad a las frequencias bajas, esto viene debido a la *ley de Weber-Fechner* que establece una relación cuantitativa entre la magnitud de un estímulo físico y cómo este es percibido. Según esta ley *"el menor cambio discernible en la magnitud de un estímulo es proporcional a la magnitud del estímulo actual"*[63], por eso para frecuencias bajas la fracción necesaria para percibir un cambio es mucho menor que para frecuencias altas, donde los cambios en frecuencia han de ser mas grandes para notar algún cambio.
El espacio de frecuencia Mel en función de la frecuencia lineal viene dado por la ecuación 4.1. Esto provoca que si por ejemplo queremos hacer una división de 24 filtros entre las

frecuencias 0 y 2000Hz, siguiendo una distribución lineal quedaría como se muestra en la figura 4.1.

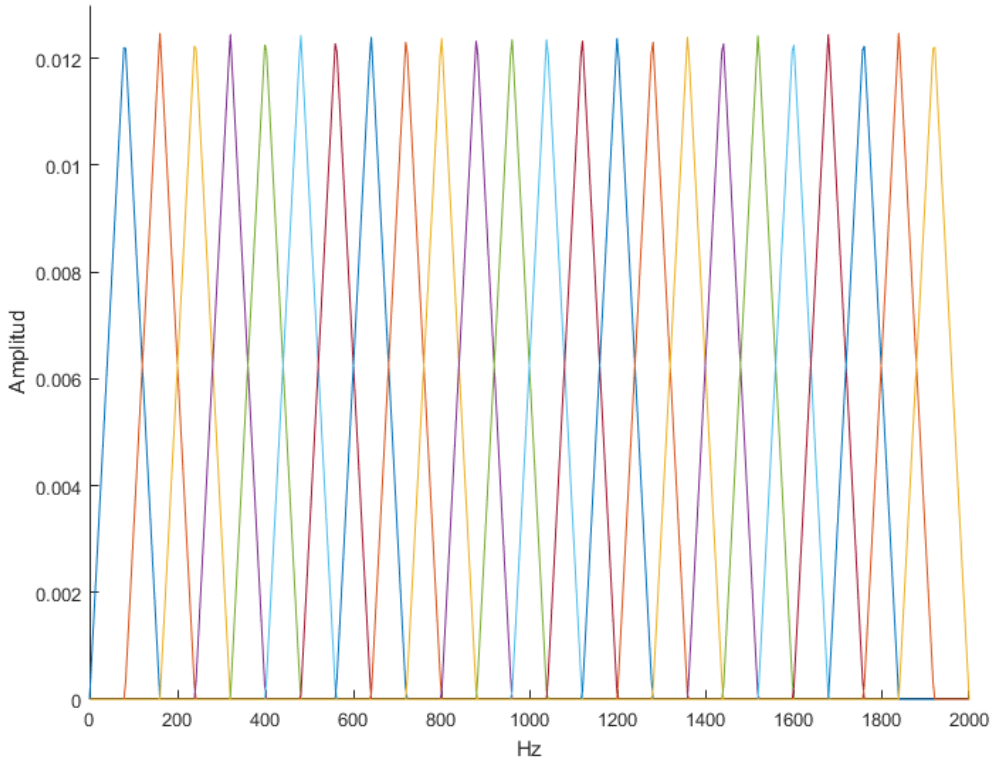

Figura 4.1 Filtros distribuidos uniformemente en frecuencia lineal.

Mientras que en el espacio de frecuencia mel quedaría tal y como se muestra en la figura 4.2.

La regla existente entre las frecuencias lineales y las mel viene dada por la ecuación 4.1

$$mel(f) = 1125 * ln(1 + f/700) \tag{4.1}$$

Como se puede observar los filtros tienen un mayor ancho de banda para frecuencias mas altas puesto que el cerebro es menos sensible en estas zonas, de esta forma se obtiene un filtrado mas parecido al que realiza el oído humano. Realizaremos el cálculo de los MFCC tal y como se indica en [9], primero realizando un preénfasis para compensar los formantes de altas frecuencias que fueron suprimidos con los mecanismos de producción del sonido de los instrumentos, mediante la ecuación 4.2:

$$M_p(n) = m(n) - 0.97 * m(n-1) \tag{4.2}$$

Donde $M_p(n)$ es la salida señal con el preénfasis y $m(n)$ es la señal original.
Otro de los aspectos a tener en cuenta es el enventanado. Cuando se realiza la STFT (*Short-Time Fourier Transform*), pueden producirse cambios abruptos entre tramos contiguos debido a las discontinuidades, enventanar estos tramos puede ayudar a minimizar el efec-

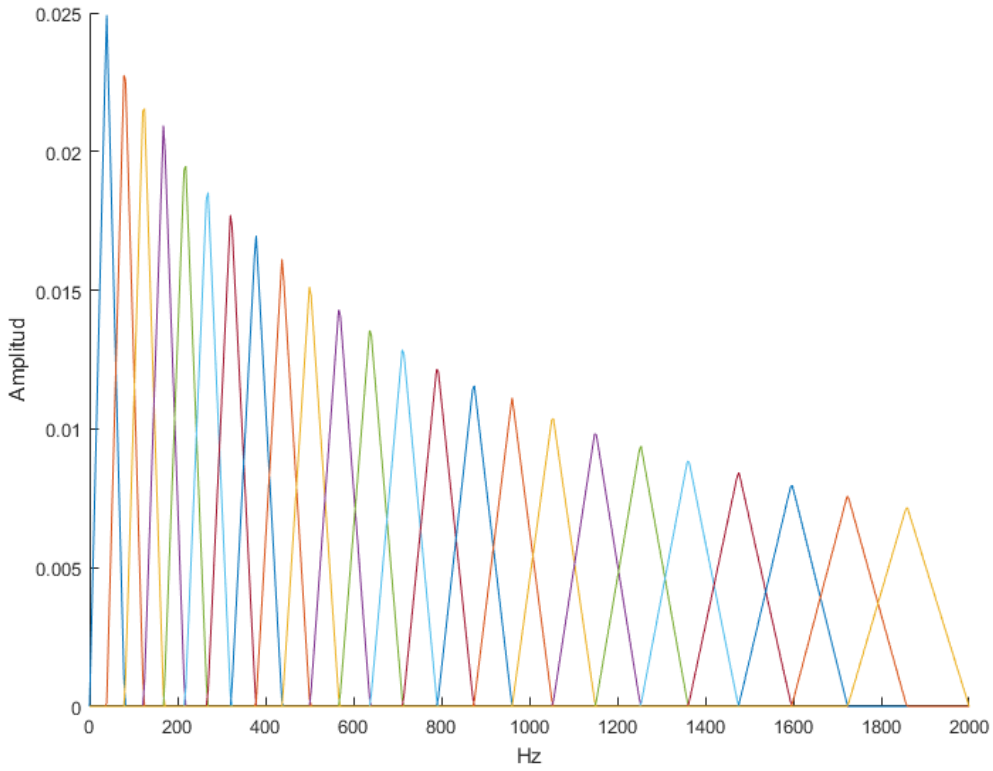

Figura 4.2 Filtros distribuidos uniformemente en mel visto en frecuencias lineales.

to que tienen las discontinuidades, en concreto nosotros lo haremos con la ventana de *Hamming*, mediante $M_p(n) * W(n)$, siendo $W(n)$ la ventana de *Hamming* definida por la ecuación 4.3:

$$W(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right) \tag{4.3}$$

Una vez enventanada la señal, se hará pasar por un banco de 24 filtros distribuidos uniformemente en el espacio definido por 4.1 entre 0 Hz y 11000 Hz, siendo estos filtros triangulares. Para finalmente obtener los coeficientes, aplicamos la transformada discreta del coseno (de ahora en adelante DCT). Esta se usa por su propiedad para compactar la energía. Los coeficientes se obtienen según la ecuación 4.4:

$$C_i(m) = \sum_{n=0}^{M-1} S_i(n)\cos\left[\pi m\left(\frac{n-0.5}{M}\right)\right], 0 \leq m \leq M \tag{4.4}$$

Donde M es el número de filtros triangulares pasobanda, $S_i(n)$ es el *cepstrum* del logaritmo de la salida del banco de filtros y $C_i(m)$ el coeficiente MFCC del i^{esimo} tramo.
A estos resultados se les realizará la media y la varianza para todos los *frames* quedando pues la media de los 24 coeficientes, finalmente nos quedaremos con los 13 primeros valores de estas medias y varianzas, que se usarán como características para nuestro sistema.

4.1.2 Centroide espectral

Define el centro de gravedad del espectro. Indica lo "brillante" que es un instrumento y viene determinado por la ecuación 4.5:

$$SC = \frac{\sum_{k=1}^{K} X(f_k) f_k}{\sum_{k=1}^{K} X(f_k)} \tag{4.5}$$

Donde $X(f_k)$ es la transformada discreta de Fourier de la señal $x(n)$ en el tiempo y f_k es el k^{esimo} *bin* de frecuencia.

4.1.3 Flujo espectral

Es una medida que permite determinar la cantidad de variación que existe entre el espectro resultante de un *frame* y su siguiente, se mide como la diferencia al cuadrado entre espectros de frecuencias normalizados sucesivos, que se obtiene de la relación 4.6.

$$SF = \sum_{k=2}^{K} \left(|X_k(f) - X_{k-1}(f)|\right)^2 \tag{4.6}$$

De aquí volvemos a obtener la media y la varianza a través de todas las ventanas. Además hemos añadido otra característica basada en el flujo espectral y el centroide espectral, el centroide del flujo espectral.

4.1.4 Centroide del flujo espectral

En vez de calcular el cuadrado de la norma, obtenemos un vector de espectros de frecuencia como la magnitud de la resta dos ventanas $X_k(f)$ consecutivas, a las cuales les calculamos el centroide, de esta forma sabremos cuales son las zonas del espectro mas propensas a sufrir cambios, de este nuevo centroide obtenemos la media y la varianza a lo largo del vector de diferencias entre espectros de frecuencias de muestras consecutivas. Puede expresarse como:

$$CSF = \frac{\sum_{k=2}^{N} f\,|X_k(f) - X_{k-1}(f)|}{\sum_{k=2}^{N} |X_k(f) - X_{k-1}(f)|} \tag{4.7}$$

4.1.5 Caída espectral

La caída espectral o *spectral roll-off* es una medida de la forma de una señal y se define en 4.8 como el *bin M* de frecuencia bajo el cual está concentrado el 85 % de la distribución de magnitud del espectro. La definimos según [10]:

$$Spectral\ roll-off: \sum_{k=1}^{M} X[f_k] = 0.85 * \sum_{k=1}^{N} X[f_k] \tag{4.8}$$

Donde $X[f_k]$ es la magnitud del k^{esimo} bin de frecuencia de la DFT de $x[n]$, cuya DFT contiene N *bins* de frecuencia.

4.1.6 *Spectral Skewness*

Conocido en español como sesgo espectral, en estadística es una medida de la asimetría de una función de distribución alrededor de su media. La oblicuidad se calcula como el tercer momento estándar, y viene dado por la ecuación 4.9:

$$\gamma_1 = E\left[(\frac{X-\mu}{\sigma})^3\right] = \frac{\mu_3}{\sigma^3} = \frac{E\left[(X-\mu)^3\right]}{(E\left[(X-\mu)^2\right])^{\frac{3}{2}}} = \frac{\kappa_3}{\kappa_2^{\frac{3}{2}}} \tag{4.9}$$

Donde μ es la media, μ_3 es el tercer momento central, σ la desviación estándar, E es el operador de la esperanza matemática y κ_t son los t^{esimos} cumulantes. Obtenemos la media y la varianza de todos las ventanas realizadas. En función del *skewness* el espectro será simétrico o asimétrico con mayor energía hacia la izquierda (menor frecuencia) o hacia la derecha (mayor frecuencia) del centroide. Si $\gamma_1 > 0$ la distribución del espectro es asimétrica hacia la derecha, si $\gamma_1 = 0$ la distribución del espectro es simétrica y por último, si $\gamma_1 < 0$ la distribución del espectro es asimétrica hacia la izquierda.

4.1.7 *Spectral Kurtosis*

Conocida en español como *curtosis*, en estadística es una medida de la concentración de valores alrededor de la media de una función de distribución. Cuanto mayor es la curtosis, mayor será la concentracion de valores cerca de la media de la distribución (pico) y muy lejos de la misma (colas) al mismo tiempo que existe una relativamente menor frecuencia de valores intermedios (hombros), en resumen indica lo plano o "picudo" que es el espectro en función de los valores obtenidos, respecto a una distribución normal. Este es el cuarto momento estandarizado y viene dado por 4.10. Obtenemos la media y la varianza de todos las ventanas realizadas.

$$\beta_2 = \frac{\mu_4}{\sigma^4} \tag{4.10}$$

Donde μ_4 es el cuarto momento central y σ la desviación estándar. Si $\beta_2 = 2/3$ la distribución es normal, si $\beta_2 > 2/3$ la distribución es más picuda, si $\beta_2 < 2/3$ la distribución es más plana.

4.1.8 Spectral Flatness

Se define como el ratio que existe entre la media geométrica y la aritmética de un array. Suele medirse en decibelios y proporciona un método para cuantificar como de "ruidoso" o "tonal" es un sonido. Mide pues la cantidad de picos o estructuras especialmente resonantes en un espectro que lo diferenciarían por ejemplo de un ruido blanco. Un valor muy alto (siendo aprox. 1 para el ruido blanco) implica una distribución equitativa de la potencia a lo largo del espectro. En cambio cuanto mas cercana es a 0, indica que la potencia espectral está concentrada en un relativo número de bandas mas pequeñas. Su ecuación es 4.11.

Obtenemos la media y la varianza de todos las ventanas realizadas.

$$Flatness = \frac{\sqrt[K]{\prod_{k=0}^{K} X(f_k)}}{\frac{\sum_{k=0}^{K} X(f_k)}{K}} \tag{4.11}$$

4.2 Temporales

Zero Crossing Rate

La tasa de cruces por cero (ZCR) es una forma de medir lo ruidosa que es una señal. Se divide y enventana como se menciona en 4.1.1, para cada una de esas ventanas se calcula la ZCR como en 4.12, siendo la media y la varianza de esta las que se utilizarán finalmente como características para el clasificador.

$$ZCR = \frac{1}{N} \sum_{0}^{N-1} |sgn\,[m(n)] - sgn\,[m(n-1)]| \tag{4.12}$$

Donde $m(n)$ es la n muestra de la señal enventanada m, $m(n-1)$ la muestra anterior y N el número total de muestras en cada tramo.

4.2.1 BPM

Son los *beats per minute* o el *tempo* de la canción. Géneros como el reggaeton son conocidos por llevar un tempo mas lento que por ejemplo la salsa o la bachata, de hecho el reggaeton y la música electrónica mas comercial son claros ejemplos de como los productores han adaptado "estándares" de tempo para evitar que sus canciones varíen el *pitch* conforme se cambia el tempo para coincidir con el del siguiente tema en la mezcla, en la música electrónica este tempo estándar esta alrededor de 128bpm mientras que en el reggaeton se sitúa alrededor de 94bpm. Para los otros estilos es muy diferente, la bachata puede variar aunque por lo general la bachata dominicana (o clásica para ser mas justos) suele ser un tanto mas acelerada que la bachata moderna, una suele oscilar sobre los 140-160 bpm (dominicana) mientras que la moderna suele estar alrededor de unos 120-130 bpm. El caso más interesante es el de la salsa, suele escribirse como la mayoría de la música hoy en día en 4/4, no obstante la estructura de acentos y ritmos no se repiten cada 4 tiempos si no cada 8 (véase figura 4.4), y las frases musicales por ejemplo de "llamada y respuesta" serán siempre múltiplos de 2 compases, pero jamás realizaran llamada y respuesta en sólo dos compases (4.3), esto desencadena en que los algoritmos de estimación de tempo detecten la salsa a la mitad de tempo de lo que se escribe en la partitura, por lo que aunque en salsa los mínimos están alrededor de 160bpm en nuestras características aparece como 80bpm.

El tempo se detecta con la función *beat tracker degara*, esta función enventana la señal de audio con 2048 muestras y 1024 muestras de solape para posteriormente calcular la diferencia entre cuadros sucesivos del espectro complejo de frecuencias, un método originario de [64]. El método podría resumirse de la siguiente forma, se normaliza la señal resultante de cada ventana mediante una media adaptativa y corrección de media onda.

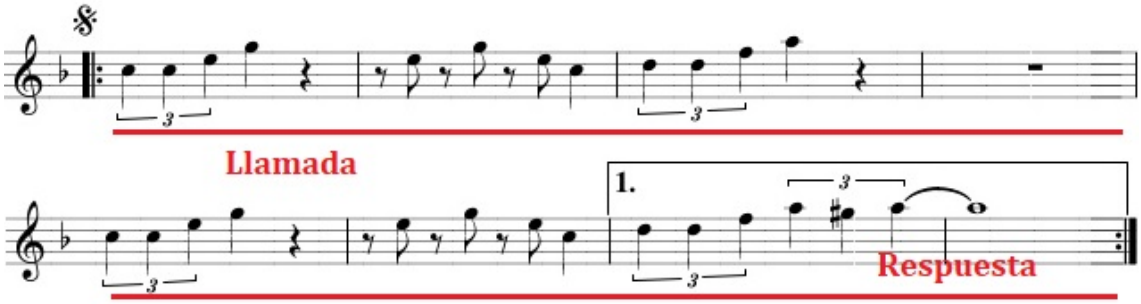

Figura 4.3 Extracto de Rebelión de Joe Arroyo.

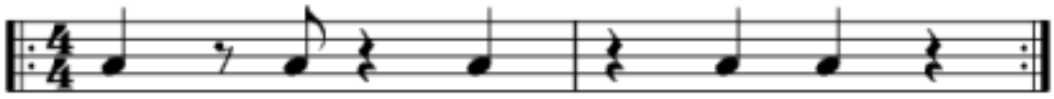

Figura 4.4 Patrón rítmico de la clave en salsa.

Se calcula la autocorrelación para descartar información relacionada con la fase, de esta forma se obtiene la información relacionada con la periodicidad del mismo. La función *beat tracker degara* devuelve un vector con los segundos en los que ocurre cada pulso, de esta forma obteniendo la media de la diferencia entre pulsos consecutivos y dividiendo 60 entre la misma podemos obtener la cantidad de pulsos por minuto o *BPM*. Sea el vector x de n pulsos, tal que $[x_1, x_2, \ldots, x_n]$ son los distintos pulsos consecutivos, entonces:

$$BPM = \frac{60}{\frac{\sum_{k=2}^{n}(x_k - x_{k-1})}{n}} \tag{4.13}$$

4.2.2 RMS

La media cuadrática o *Root mean square* (RMS) es la medida estadística de la magnitud de una cantidad variable. El nombre deriva del hecho de que es la raíz cuadrada de la media aritmética de los cuadrados de los valores del vector de la señal. En este caso, el vector contiene las muestras de la señal de audio. Se puede expresar para una variable discreta como:

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_N^2}{N}} \tag{4.14}$$

Siendo en nuestro caso x_i cada uno de los *frames* en los que separamos la canción. Se extrae como característica la media y la varianza de la media cuadrática a lo largo de todos los *frames* extraídos para cada canción.

4.3 La transformada fraccional de Fourier

Esta característica se encuentra fuera de las secciones 4.1 y 4.2 porque la transformada fraccional de Fourier transforma cualquier señal en un dominio entre tiempo y frecuencia. Se usa en diseño de filtros, análisis de señales, estimación de fase, reconocimiento de patrones y en este caso como característica para clasificar géneros musicales [10][9].

4.3.1 Definición

Puede pensarse como la transformada de Fourier a la n-ésima potencia no necesariamente siendo n un entero. Este es el caso que nos interesa, puesto que cuando no es un entero es cuando se generan los espacios intermedios entre tiempo y frecuencia.

Sea la definición de transformada de Fourier de una función $x(t)$ como :

$$\mathscr{F}\{x(t)\} = y(f) \tag{4.15}$$

Entonces se define:

$$\mathscr{F}^{\alpha}\{x(t)\} = y_{\alpha}(f) \tag{4.16}$$

siendo $\mathscr{F}^{\alpha}$ la "transformada fraccional de Fourier de orden α" siendo α el "orden fraccional". Puede escribirse como:

$$\mathscr{F}_{\alpha}[f](u) = \sqrt{1-i\cot(\alpha)}e^{i\pi\cot(\alpha)u^2}\int_{-\infty}^{\infty} e^{-i2\pi\left(\csc(\alpha)u\,x-\frac{\cot(\alpha)}{2}x^2\right)}f(x)dx \tag{4.17}$$

No obstante también es muy común encontrar la definición de la misma en forma de convolución con un *kernel* [65] que será la que usaremos a partir de ahora:

$$f_a(u) = \int_{\infty}^{\infty} K_a(u,u')f(u')du' \tag{4.18}$$

$$K_a(u,u') = A_{\alpha}exp\left[i*\pi(\cot(\alpha u^2) - 2\csc(\alpha uu') + \cot(\alpha u'^2))\right] \tag{4.19}$$

$$A_{\alpha} = \sqrt{1-i\cot(\alpha)} \tag{4.20}$$

$$\alpha = \frac{a*\pi}{2} \tag{4.21}$$

Nota: en 4.16 α hace referencia al orden de la tranformada mientras que en 4.18 se usa a para referirnos al mismo y α se usa en cambio en 4.19 y 4.21 como el ángulo que forma el dominio intermedio con el dominio del tiempo (Véase la figura 4.5)

4.3.2 Propiedades

La transformada satisface las siguientes condiciones

$$\mathscr{F}^0\{x(t)\} = x(f) \tag{4.22}$$

$$\mathscr{F}^1\{x(t)\} = y(f) \tag{4.23}$$

Además también posee una importante serie de propiedades:

Índices aditivos

$$\mathscr{F}^\beta\{\mathscr{F}^\alpha\{x(t)\}\} = \mathscr{F}^{\beta+\alpha}\{x(t)\} \tag{4.24}$$

Inversa

$$(\mathscr{F}^\alpha)^{-1} = \mathscr{F}^{-\alpha} \tag{4.25}$$

Linearidad

$$\mathscr{F}^a\left[\sum_k b_k f_k(u)\right] = \sum_k b_k\left[\mathscr{F}^a f_k(u)\right] \tag{4.26}$$

Por lo tanto podría considerarse con 4.24 a la transformada de orden α como aplicar la transformada de Fourier un número α de veces consecutivas, de lo cual deducimos junto a 4.22, 4.23 y 4.25 que:

$$\mathscr{F}^2 = x(-t) \tag{4.27}$$

$$\mathscr{F}^4 = x(t) \tag{4.28}$$

4.3.3 Dominios

El concepto mas importante de la transformada clásica de Fourier es el de "dominio de la frecuencia", donde las señales resultan mas intuitivas de comprender, es por ello que cabría preguntarse cual es el sentido o donde reside el espacio *tiempo-frecuencia*. Resulta mas fácil de entender si se atiende a la figura 4.5 que muestra en un plano los dominios t (tiempo) y f (frecuencia). Los ejes oblicuos bajo el ángulo α constituyen los dominios intermedios entre la frecuencia y el tiempo, donde el a^{esimo} orden y α están relacionados según la ecuación 4.21, lo cual es consistente con las propiedades 4.24, 4.22 y 4.23. Además podemos ver un ejemplo de aplicación para filtrado de señales mediante la rotación del espacio tiempo en 4.6. La progresiva transformación entre tiempo y frecuencia puede verse en 4.7

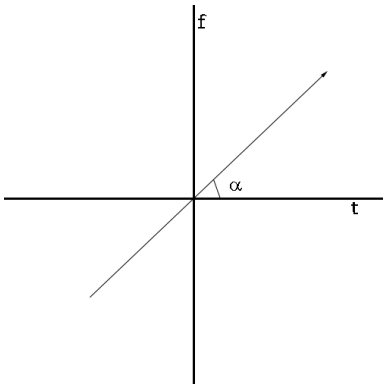

Figura 4.5 Plano tiempo-frecuencia.

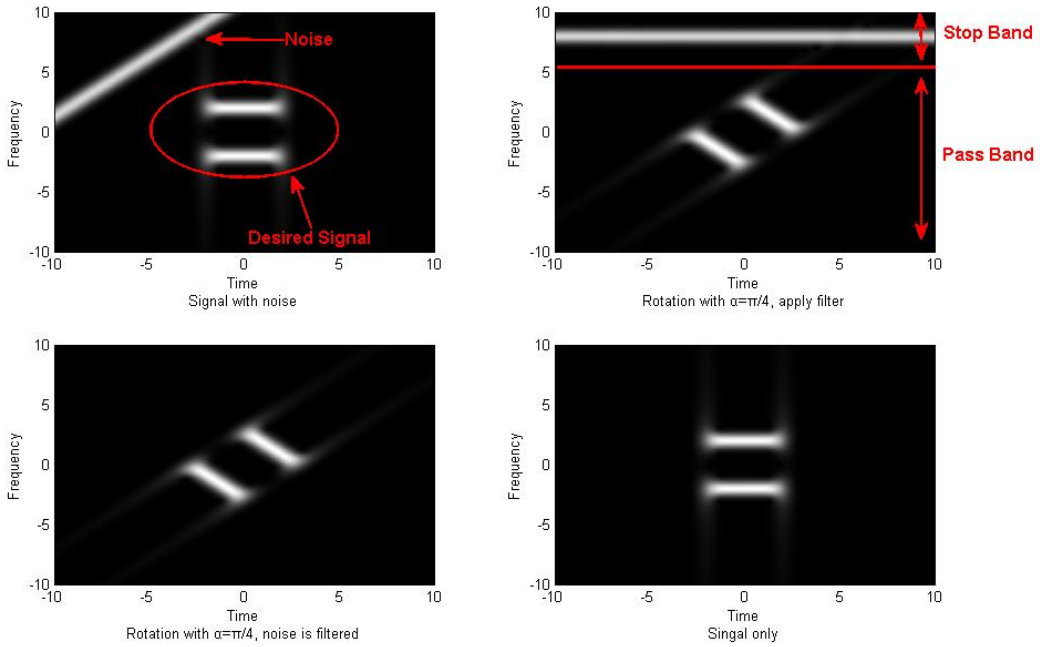

Figura 4.6 Ejemplo de aplicación de la FrFFT [66].

4.3.4 Características

Una vez calculada la transformada de Fourier fraccional procedemos a extraer los MFCC con el proceso explicado en 4.1.1, y de estos volveremos a calcular la media y la varianza a lo largo de todas las ventanas. El valor que permite la detección óptima se obtenido mediante un estudio experimental cuyo el valor se mostrará en el apartado de resultados.

4.4 Resumen de Caracetrísticas

Las características usadas son el total 71, y se resumen en la tabla 4.1.

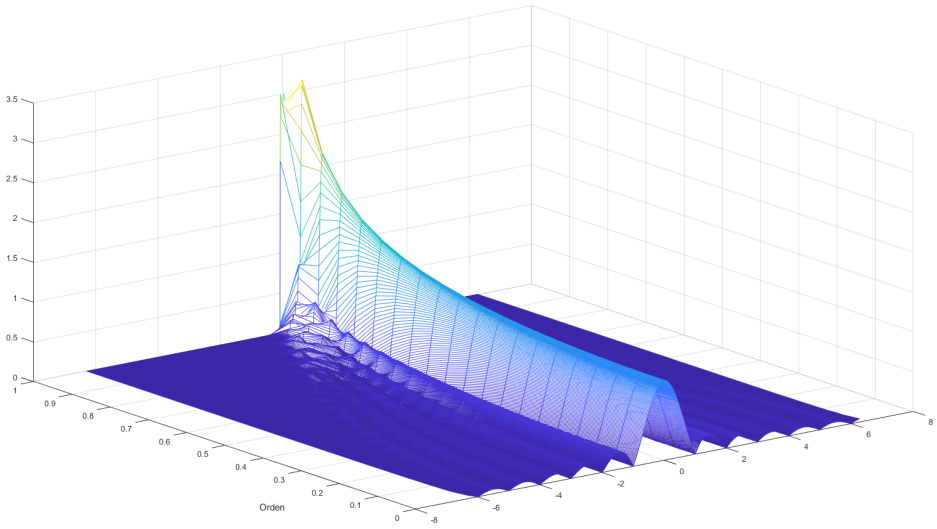

Figura 4.7 Transformada fraccional de Fourier desde orden 0 a 1 de una señal Sinc().

Tabla 4.1 Resumen de características.

Num. Características	Tipo	Características
40	Tímbricas	Varianza y media de los 13 primeros MFCC, varianza y media del centroide espectral, varianza y media del flujo espectral, varianza y media del centroide del flujo espectral, varianza y media de la caída espectral (o *Roll-off*), varianza y media del sesgo espectral, varianza y media de la curtosis espectral, varianza y media de la *spectral flatness*
5	Temporales	BPM o tempo, media y varianza de la tasa de cruce por cero, media y varianza de la media cuadrática
26	FrFFT	Media y varianza de los 13 primeros FrMFCC

5 Resultados

Sin desviarse de la norma, el progreso no es posible.

FRANK ZAPPA

Este capítulo explica cómo se han obtenido los resultados, los métodos finalmente utilizados y su posterior evaluación.

5.1 Generación de Resultados

El procedimiento para extraer los resultados se puede observar en el esquema presentado en la figura 5.1.

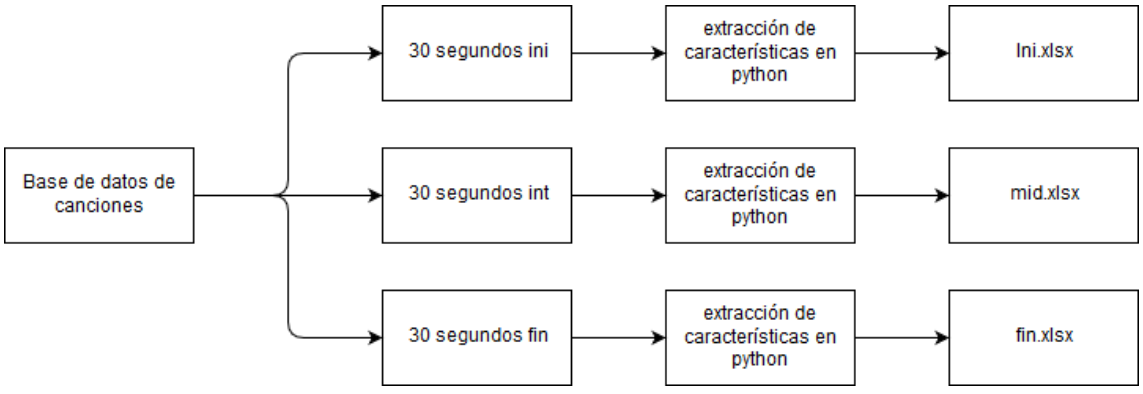

Figura 5.1 Proceso de extracción de características.

Y el procedimiento para extraer los datos de la tabla con matlab y entrenar los modelos puede verse en la figura 5.2.

Figura 5.2 Proceso de lectura de características y creación de los modelos de clasificación.

La base de datos está conformada por 597 canciones de las cuales 129 son de bachata, 98 son de chachachá, 118 son de merengue, 115 canciones son de reggaetón y 137 de salsa. La elección de estas canciones se ha basado en la diversidad de los artistas, de manera que el género no venga dado por la interpretación propia de un artista o grupo en concreto del género, proporcionando así una visión mas amplia.

Extraemos 30 segundos del inicio, el intermedio y el final de cada audio, los cuales serán tratados por un código en python el cual extrae todas las características exceptuando las de la transformada fraccional de Fourier, las cuales se realizan a posteriori para calcular la α óptima para el resultado. Cuando termina de extraer las características de todos los fragmentos genera tres tablas de *Excel*, una para cada parte de las canciones. La tabla contiene todas las características espectrales y temporales, los nombres de cada característica así como el título de la canción y el género al que pertenece, como puede observarse en la figura 5.3.

	A	B	C	D	E	F	G	H
1	Genre	SongName	meanmfcc1	meanmfcc2	meanmfcc3	meanmfcc4	meanmfcc5	meanmfcc6
2	Chachacha	16. Green eyes.m4	-746,211669921875	114,949272	14,7274323	-11,492189	-4,9330077	2,52166486
3	Chachacha	11. Cha-cha-in-g.	-780,761657714844	68,4316711	12,143033	7,82994223	-2,4397547	-2,397253
4	Chachacha	16 - celia cruz - o	-715,41015625	84,4155273	7,98004007	16,707283	9,77196789	-4,8441892
5	Chachacha	Pete_Rodriguez -	-651,602600097656	96,2783127	-23,130539	13,6023006	-3,9122953	0,18761453
6	Chachacha	16. El bajo.m4a	-772,060668945313	86,543335	32,1395378	20,6496181	9,43413258	9,59973431
7	Chachacha	15. Cha-cha-stroll	-733,875732421875	69,1910934	-2,9869847	13,1093388	-2,7126615	0,50808346
8	Chachacha	16. L' etranger au	-671,636474609375	51,5577393	-12,462378	18,848238	-9,7279482	4,84701633
9	Chachacha	22. Silencio.m4a	-739,332336425781	97,1904449	8,73820019	-9,3538284	-4,2105002	-0,8406776
10	Chachacha	11. El organillero.	-736,462524414063	93,9034882	-1,587819	14,0533752	1,91287613	-14,176708
11	Chachacha	05. Nature boy.m4	-755,021789550781	67,5421906	-3,602421	2,2932055	-6,484992	-3,7601459
12	Chachacha	03. Senor Juez.m4	-744,3173828125	95,4717865	1,24391937	12,4368563	7,64968204	-11,859575
13	Chachacha	09. Frenesi.m4a	-731,039123535156	71,1950836	-9,7100258	7,6778574	-3,6606386	-3,3410544
14	Chachacha	11. Mona Lisa.m4	-678,341247558594	83,1120834	-54,17897	3,75387597	-2,2061079	-4,4355702
15	Chachacha	14. La gloria eres	-758,175231933594	88,5596542	10,728879	19,9465446	5,16845798	-5,9863582
16	Chachacha	01. Lindo cha cha	-689,08740234375	98,2986984	-8,4757748	5,62214231	-4,0311537	-6,1931925
17	Chachacha	10. Trumpet Cha	-767,919982910156	46,4270172	-4,477879	3,31106257	-4,429378	3,49456453
18	Chachacha	Cheo_Feliciano - C	-710,361755371094	76,1954041	14,3037815	21,4209175	-2,3018854	3,90817809
19	Chachacha	06. Habanero.m4	-700,639831542969	102,346451	-6,9729548	-0,7485394	-3,7025776	-3,2179799
20	Chachacha	20. Memoria a ch	-722,120727539063	101,904251	-14,215187	8,98292542	0,57910872	-13,586367
21	Chachacha	23. The Sheik of A	-729,329772949219	73,1802979	-13,878598	-0,9698303	-2,4753411	-3,2596681
22	Chachacha	18. El Palo.m4a	-680,952941894531	88,358284	-18,047558	9,91750908	-2,5058851	-4,5335588
23	Chachacha	10. Yo tengo una	-747,962951660156	90,1111679	3,12663102	14,3795691	5,1634264	-9,9219027
24	Chachacha	01. El bodeguero.	-732,863586425781	102,484695	-0,1068017	12,2869816	8,49381733	-2,0049775
25	Chachacha	24. Esto es Felicid	-749,091491699219	72,094696	-6,3875709	16,3330231	1,4313004	4,29570818
26	Chachacha	03. Patricia.m4a	-749,137756347656	84,0931396	2,13002157	4,13133478	-3,1594796	-1,6551844
27	Chachacha	04. Los fantasmas	-721,724670410156	104,323257	-6,2710671	11,0677185	7,86033058	-9,9652109
28	Chachacha	05. Corazon de M	-794,299194335938	66,3796082	10,0456219	6,60920048	6,15625429	7,32782555

Figura 5.3 Captura del *Excel* de salida del algoritmo que extrae las características temporales y espectrales.

La extracción de las características basadas en la Transformada Fraccional de Fourier

se realiza mediante otro código extraído de *GitHub* [67]. No obstante como la carga computacional de este cálculo es muy elevada realizaremos una primera aproximación con las características temporales y espectrales para determinar cuál de las tres partes de cada canción *inicio, intermedio, final* produce una mayor tasa de aciertos, y en base a ello extraeremos los *FrMFCC* (*Fractional Mel Frequency Cepstral Coefficients*) con varios valores de α para esa parte, con el objetivo de buscar el valor de α que permite una mejor clasificación.

5.1.1 Aprendizaje supervisado

Esta parte es la principal del trabajo, pues es la que nos proporciona mejores resultados y además nos servirá de guía para el estudio de aprendizaje no supervisado. Caben distinguir dos partes aquí, la extracción de características en *python* y la clasificación usando la aplicación de Matlab *classification learner* (véase figura 5.4).

Python

La extracción de características en *Python* se ha realizado con la versión 3.4 del mismo usando la librería *essentia* [68] realizada por la *Universitat Pompeu Fabra* y el grupo de investigación *Music Technology Group*. Este entorno se ha construido bajo el sistema operativo *Debian Stretch* para mayor compatibilidad. Así mismo para guardar los datos en formato *Excel* se ha utilizado *openpyxl* [69].

Para el cómputo de la transformada fraccional de Fourier se ha utilizado el paquete de código abierto del repositorio [67]. Una vez determinado cual de las tres secciones de cada canción presentaba un mayor potencial discriminante. Se realió un cálculo de 8 α (orden de la transformada) equidistantes en el intervalo $[0.1, 0.9]$ siendo 0.1 el primer valor a comprobar y 0.9 el último, así se comprobó cual de ellos arrojaba mejores resultados en la clasificación.

Matlab

La herramienta *Classification Learner* permite realizar un barrido por una serie de algoritmos de aprendizaje supervisado de manera rápida y automática, así como afinar los parámetros con el fin de buscar el algoritmo que genera mejores resultados para el campo a estudiar. En este caso nuestro clasificador óptimo se compondrá de un *ensemble* mediante el método de *Random Subspaces*, utilizando subespacios de 36 dimensiones y compuesto de 30 clasificadores de discriminantes "pseudo-lineales". Este tipo de discriminante es similar al explicado anteriormente exceptuando que la inversa de la matriz de covarianza compartida por todas las clases es calculada con el algoritmo de la pseudoinversa [51][70].

5.1.2 Aprendizaje no supervisado

Para el aprendizaje no supervisado solo se ha usado Matlab, a posteriori del estudio de aprendizaje supervisado mediante la función *kmeans* de Matlab, que dado una serie de puntos y el número de *clusters* devuelve a cual de ellos pertenece cada punto. Para reducir

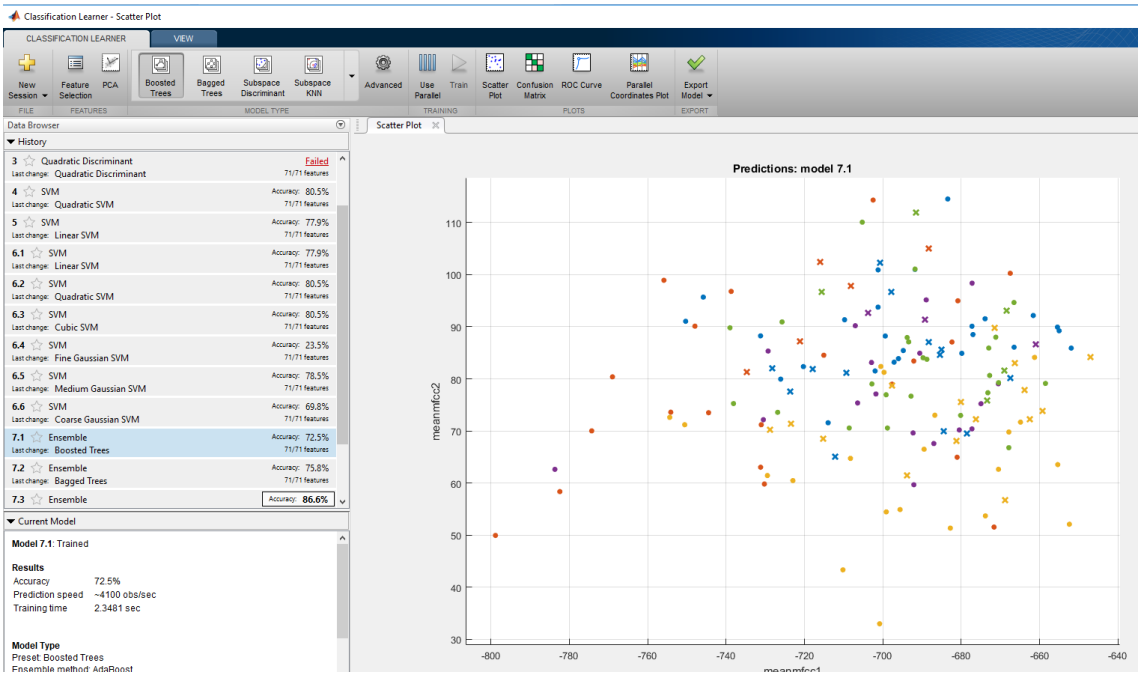

Figura 5.4 Captura de pantalla de *Classification Learner*.

y optimizar el espacio en el cual se realizan los *clusters* se ha utilizado el algoritmo *t-Distributed Stochastic Neighbor Embedding* [71] [72], y se buscan los parámetros óptimos mediante el método de *rand index* que se explica en el siguiente apartado.

5.2 Valoración de los resultados

En esta sección procederemos a valorar los resultados obtenidos y cuáles han sido los criterios elegidos para evaluar los modelos.

Aprendizaje supervisado

El aprendizaje supervisado ha sido el más preciso como cabía esperar. Mediante el método descrito en 5.1.1 se ha alcanzado una tasa de aciertos del 88.6 %, otros métodos como las máquinas de soporte vectorial también han obtenido un buen resultado, aunque no tan alto como el conseguido con el *ensemble* (véase la tabla 5.1). El entrenamiento y la posterior evaluación del modelo se ha realizado dividiendo aleatoriamente el set total de datos en dos partes, una de un 70 % del total para el entrenamiento del modelo y el 30 % restante se usa para la verificación del mismo. La matriz de confusión resultante de la evaluación de este modelo se puede ver en la figura 5.5.

La sección de las canciones que mejor permite clasificarlas es la sección intermedia, esto puede ser debido a que al principio y al final de una canción falten elementos que ayuden a discriminar mejor las distintas voces de una canción.

Los resultados mostrados en la tabla 5.1 son sin añadir los *frMFCC*, se han obtenido realizando el proceso de separación de muestras para entrenamiento y *test* 4 veces, y obteniendo la media aritmética de la tasa de resultados.

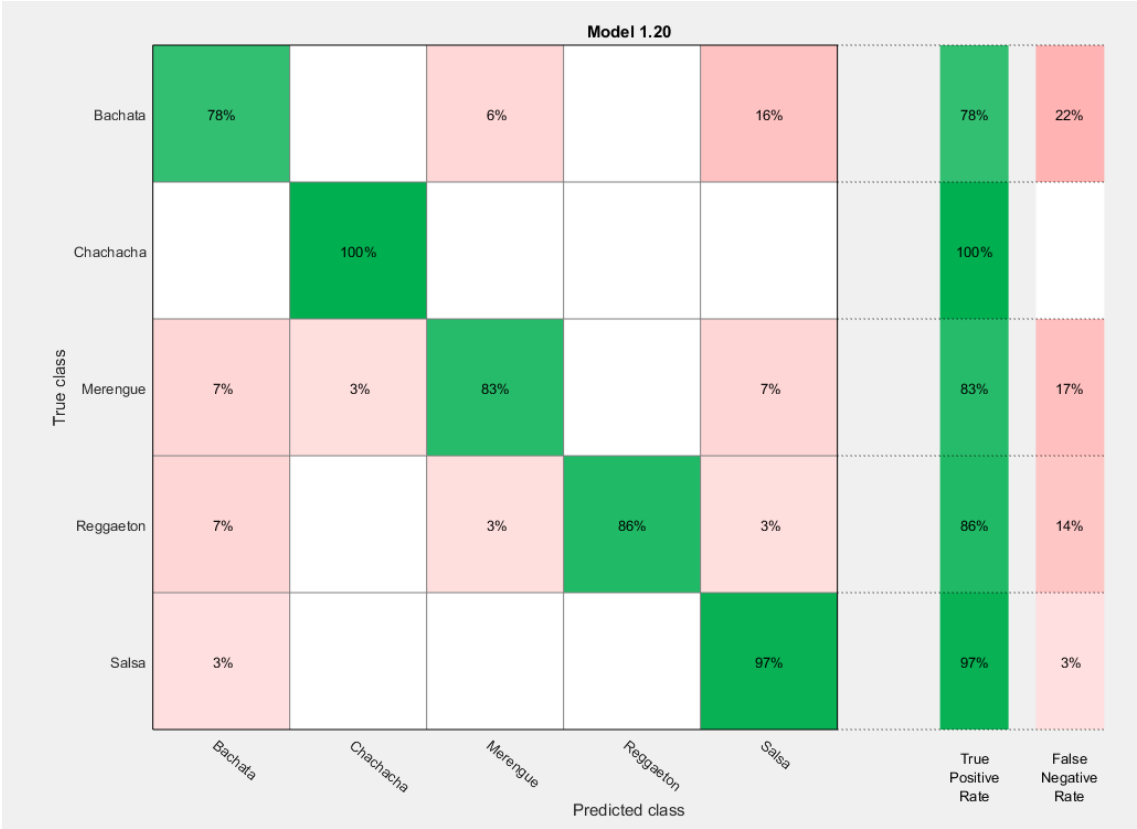

Figura 5.5 Matriz de confusión para el modelo de *Ensemble*.

Tabla 5.1 Resultados obtenidos previamente a la adición de los *frMFCC*.

	Medium Tree	*Quadratic SVM*	*Cosine KNN*	*Subspace Discriminant*
Inicio	59.95 %	**76.45** %	70.15 %	76.125 %
Mitad	66.27 %	82.33 %	72.45 %	**85.675** %
Final	66.45 %	80.35 %	73.95 %	**81.07** %

Un caso a destacar es que aunque el método de *subspace discriminant* es superior en los casos medio y final, en inicio lo es *quadratic svm*, no obstante con tasas mucho menores a las que se obtienen con *subspace discriminant* en los casos medio y final.

Una vez realizamos el procedimiento experimental para obtener la α óptima, determinamos que esta es $\alpha = 0.7857$. En la tabla 5.2 se pueden observar los resultados obtenidos con los *frMFCC* para el óptimo y los dos valores α más cercanos al mismo, mediante la media de 4 iteraciones al igual que en la tabla 5.1.

Tabla 5.2 Resultados obtenidos posteriormente a la adición de los *frMFCC*.

	Medium Tree	*Quadratic SVM*	*Cosine KNN*	*Subspace Discriminant*
$\alpha = 0.6714$	63.32 %	81.6 %	72.9 %	**85.375** %
$\alpha = 0.7857$	65.45 %	82.87 %	75.4 %	**86.8** %
$\alpha = 0.9$	67.67 %	83.42 %	74.45 %	**86.37** %

En la tabla 5.2 se observa como *subspace discriminant* mantiene la mejor tasa sobre los otros métodos probados.

Aprendizaje no supervisado

Para el aprendizaje no supervisado el *clustering* de *k-means* ha obtenido resultados notablemente inferiores a los de el aprendizaje supervisado. El entrenamiento se ha realizado utilizando la función *kmeans* de Matlab, usando el parámetro "Replicates", el cual indica el número de veces a repetir el proceso de *clustering* variando en cada una de estas repeticiones la posición inicial de los centroides para evitar así que la solución converja a un mínimo global. La evaluación de estos resultados no es tan sencilla como en el aprendizaje supervisado puesto que aquí no existen etiquetas que ayuden a comprobar directamente los valores que estos deberían tener, no obstante si que existen métodos como el de *rand index* [73] o el de *silhouette* [74] que permiten estimar el comportamiento de los *clusters*.

En el caso de *rand index* permite conocer el parecido entre el resultado de dos procesos de *clustering*, el procedimiento es el siguiente:

Sea una serie de elementos n definidos en $S = \{\sigma_1, \dots \sigma_n\}$ y dos particiones de S, $X = \{X_1, \dots, X_r\}$ una partición de r *clusters*, y $Y = \{Y_1, \dots, Y_s\}$ una partición en s *clusters*. Entonces:

- a es el numero de pares de elementos de S que están en el mismo *cluster* en X y en el mismo *cluster* en X en Y
- b es el numero de pares de elementos de S que están en diferentes *cluster* en X y en diferentes *cluster* en Y
- c es el numero de pares de elementos de S que están en el mismo *cluster* en X y en diferentes *cluster* en Y
- d es el numero de pares de elementos de S que están en diferentes *cluster* en X y en el mismo cluster en Y

queda definido *rand index* como [75]:

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{5.1}$$

De manera intuitiva $a+b$ serían el número de acuerdos entre X e Y, así como $c+d$ el número de desacuerdos entre ambos, y R un porcentaje que nos permite medir la similitud entre ambos. Como tenemos la clase a la que debería pertenecer podemos construir un "falso" índice de clusters que nos permita averiguar cuanto difiere el resultado obtenido del que correspondería obtener realmente y compararlos.

En el caso de *silhouette* lo que nos permite evaluar es la "consistencia" dentro de los *clusters*. Sea $a(i)$ la distancia media de i a todos los objetos dentro del mismo *cluster* y sea $b(i)$ la distancia (euclídea) media mas pequeña de i a todos los puntos de cualquier *cluster*

de los que i no es miembro. Definimos entonces la función *silhouette* como:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{5.2}$$

Puede verse que $s(i)$ queda definida entre -1 y 1.

Para que la función se acerque a 1 se requiere que $a(i) << b(i)$. Y dado que la primera es una medida de consistencia interna por lo que los valores pequeños son deseables, y la segunda una medida de disparidad con los *clusters* vecinos por lo que es deseable valores grandes. Es por tanto que los valores cercanos a 1 implican un buen resultado. Por otro lado para que la función se aproxime a -1 $a(i) >> b(i)$ siendo en este caso mas apropiado que estuviera en un *cluster* vecino.

Con estas herramientas en mano y el algoritmo de *t-sne* que podemo encontrar en [71], observamos que los mejores resultados se obtienen al reducir el espacio de características con t-SNE a 16, obteniendo un $R = 0.71$. Además al observar la figura *silhouette* se puede ver que los *clusters* guardan cierta consistencia.

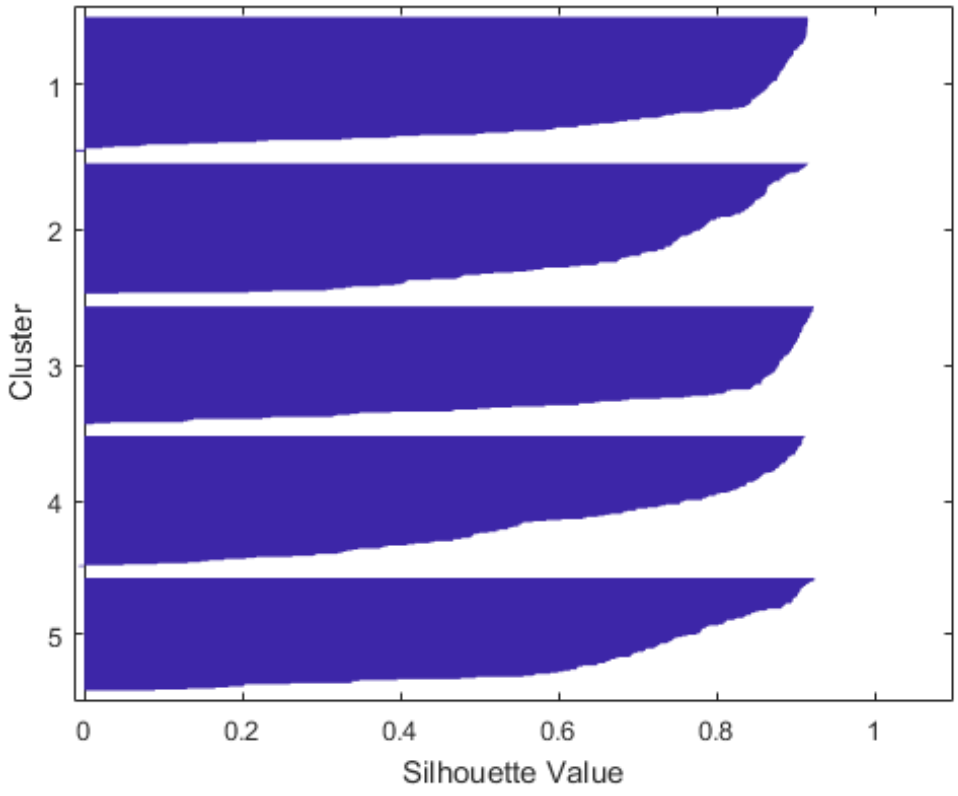

Figura 5.6 Resultado de *silhouette* para el óptimo según *rand index*.

No obstante cuando observamos la figura 5.7 podemos ver que los resultados dejan bastante que desear, a excepción del reggaeton donde hay un buen agrupamiento de canciones.

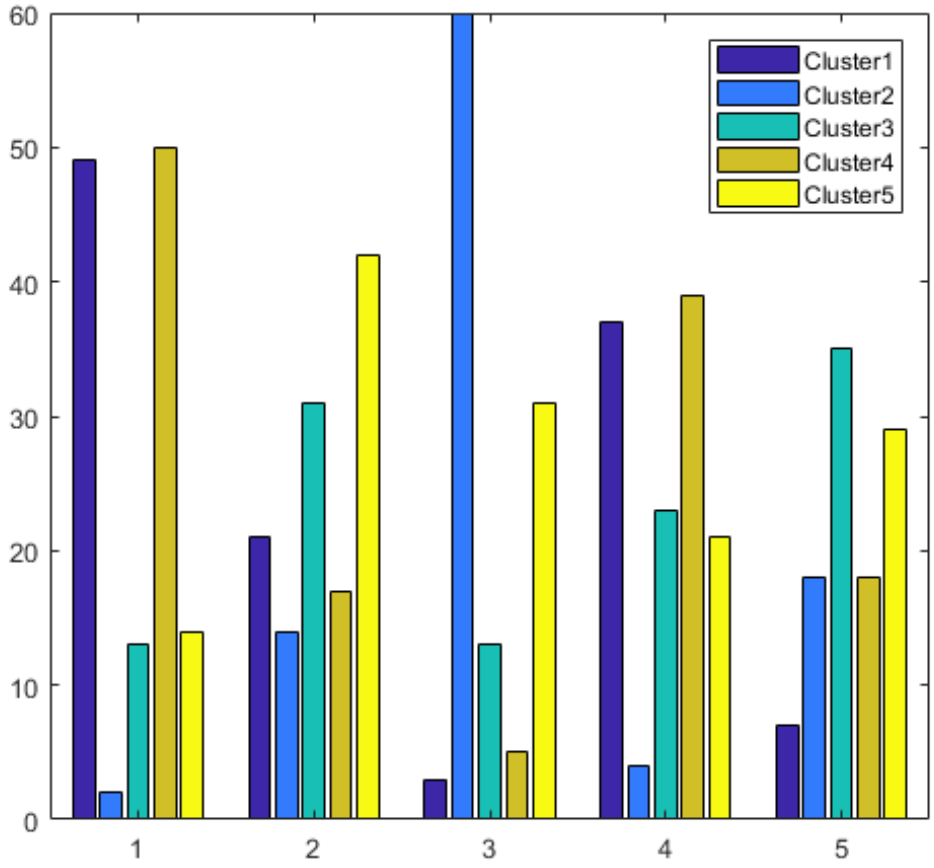

Figura 5.7 Número de puntos en cada *cluster* de (1) Bachata (2) Chachacha (3) Reggaeton (4) Salsa (5) Merengue.

Aunque sin la reducción de dimensiones, el resultado habría sido mucho peor (figura 5.8).

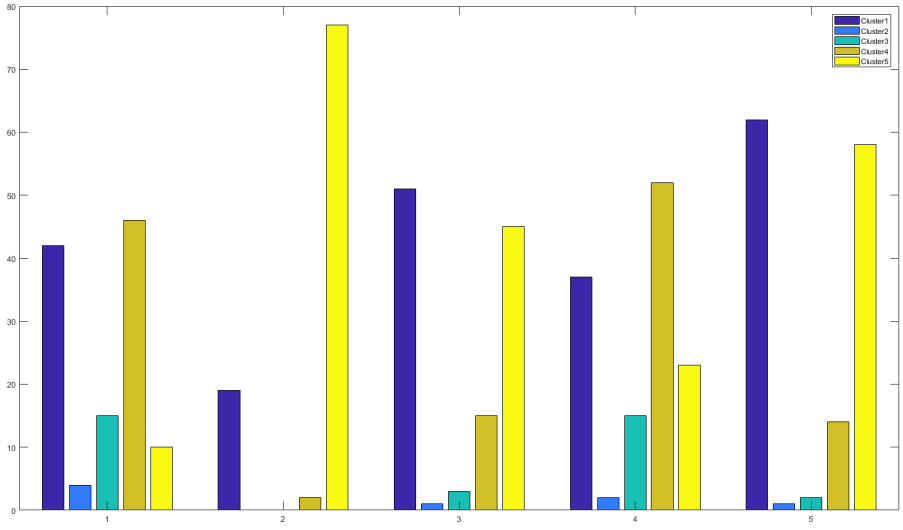

Figura 5.8 Número de puntos en cada *cluster* de (1) Bachata (2) Chachacha (3) Reggaeton (4) Salsa (5) Merengue. Caso de no reducción de dimensiones.

6 Conclusiones y líneas futuras

En este trabajo hemos visto un ejemplo de cómo se pueden clasificar 5 géneros del ámbito de la música latina y una serie de características que permiten realizar un buen acercamiento para clasificar y que proporcionan una base para futuras investigaciones .

6.1 Conclusiones

Los resultados con el aprendizaje supervisado dan a entender que las características seleccionadas son suficientes como para poder estimar con seguridad un género. Además el tiempo de procesado de canción de unos 2-3 segundos lo hace bastante aceptable como que para que pueda ser usado en un sistema de automezclado/autopinchado con cierta seguridad. La transformada fraccional de Fourier ha resultado ser un elemento útil ya que la detección aumenta en hasta un 1.125 % en el caso del *ensemble* con respecto a la mayoría de características de bajo nivel que se suelen encontrar en otros casos.

En cuanto al aprendizaje no supervisado, aunque resulta muy interesante puesto que podría tratarse de clasificar géneros musicales a ciegas, los resultados son bastante deficientes como se ven en la figura 5.7.

6.2 Líneas futuras

Los resultados obtenidos en el aprendizaje supervisado son bastante aceptables, aunque queda por observar como se comportaría el sistema si se le añaden por ejemplo las características de *kernels* basados en *gammatones* como en [27], o la inclusión de características de alto nivel como se propone en [11]. Además sería interesante ver el sistema utilizando características en tiempo real que permite extraer la librería *essentia* [68].

Aunque los resultados para el no supervisado han sido bastante malos, la cantidad de algoritmos existentes de aprendizaje no supervisado van mucho mas allá del *clustering*,

sino que existen otras técnicas como el *self-organizing map* [76] o incluso dentro del mismo *clustering* hay distintas técnicas distintas a la de *k-means*, algunas muy interesantes permiten estudiar la "asimetría" que existe en el conjunto de características para dar el mejor resultado [77].

Índice de Figuras

Índice de Tablas

Bibliografía

[1] Wikipedia. (2018) Spotify. [Online]. Available: https://es.wikipedia.org/wiki/Spotify

[2] D. Techtools. Auto-mixing tracks in playlits. [Online]. Available: https://djtechtools.com/2018/03/12/spotify-now-auto-mixing-tracks-in-playlists-gets-phrasing-right-usually/

[3] SuperMixRadio.net. (2018) Historia y origen de la salsa. [Online]. Available: http://www.supermixradio.net/historia-y-origen-de-la-salsa/

[4] R. A. Española. (2018) Diccionario de la real academia española. [Online]. Available: http://dle.rae.es/?w=bachata

[5] Unesco. (2018) Merengue patrimonio cultural inmaterial de la humanidad. [Online]. Available: https://ich.unesco.org/es/RL/la-musica-y-el-baile-del-merengue-en-la-republica-dominicana-01162

[6] "Biografía de ulises francisco espaillat," 2018. [Online]. Available: https://es.wikipedia.org/wiki/Ulises_Espaillat

[7] A. Blatter, *Revisiting Music Theory: A Guide to the Practice*. Routledge, 2007.

[8] MIREX. (2018) Music information retrieval evaluation exchange. [Online]. Available: http://www.music-ir.org/mirex/wiki/MIREX_HOME

[9] D. G. Bhalke, B. Rajesh, and D. S. Bormane, "Automatic genre classification using fractional fourier transform based mel frequency cepstral coefficient and timbral features," *Archives of Acoustics*, vol. 42, no. 2, pp. 213–222, jun 2017.

[10] D. G. Bhalke, C. B. R. Rao, and D. S. Bormane, "Automatic musical instrument classification using fractional fourier transform based- MFCC features and counter propagation neural network," *Journal of Intelligent Information Systems*, vol. 46, no. 3, pp. 425–446, may 2015.

[11] C. S. J. Rodolfo Miranda, "Using simplified chords sequences to classify songs genres," *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2017.

[12] M. M. S. D. Amélie Anglade, Emmanouil Benetos, "Improving music genre classification using automatically induced harmony rules," *Journal of New Music Research*, 2010.

[13] J. N. J. P. J.-W. Jiyoung Park, Jongpil Lee, "Representation learning using artist labels for audio classification tasks," *MIREX*, 2017.

[14] B. J. Huanmin Xu, Jiancheng Lv, "Convolutional neural networks system for the mirex 2017 audio classification (trani/test) tasks," *MIREX*, 2017.

[15] J. N. C. K. A. K. Jongpil Lee, Jiyoung Park, "Cross-cultural transfer learning using sample-level deep convolutional neural netowrks," *MIREX RESULTS*, 2017.

[16] G. Seif. (2018) Deep learning vs classical machine learning. [Online]. Available: https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa

[17] Nvidia. (2018) ¿que es cuda? [Online]. Available: https://www.nvidia.es/object/cuda-parallel-computing-es.html

[18] B. W.-m. P. L. Thierry Bertin-Mahieux, Daniel Ellis, "The milliong song dataset," *ISMIR*, 2011.

[19] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pre-trained convolu-tional neural networks for music auto-tagging," *IEEE Signal Processing Letters*, 2017.

[20] GitHub. (2018) Librosa library. [Online]. Available: https://librosa.github.io/librosa/

[21] MIREX. Audio train/test classification. [Online]. Available: http://www.music-ir.org/nema_out/mirex2017/results/act/mixed_report/

[22] R. A. Kemp, C. MacAulay, and B. Palcic, "Opening the black box: the relationship between neural networks and linear discriminant functions." *Analytical cellular pathology : the journal of the European Society for Analytical Cellular Pathology*, vol. 14, pp. 19–30, 1997.

[23] W. S. Sarle. (2000) How to measure importance of inputs? [Online]. Available: ftp://ftp.sas.com/pub/neural/importance.html

[24] A. Elbir, H. O. Ilhan, G. Serbes, and N. Aydin, "Short time fourier transform based music genre classification," in *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*. IEEE, apr 2018.

[25] J. M. de Sousa, E. T. Pereira, and L. R. Veloso, "A robust music genre classification approach for global and regional music datasets evaluation," in *2016 IEEE International Conference on Digital Signal Processing (DSP)*. IEEE, oct 2016.

[26] H. L. W. L. H. Z. Z. Zeng, S. Zhang, "A novel approach to musical genre classification using probabilistic latent semantic analysis model," *Multimedia and Expo 2009. ICME 2009. IEEE International Conference on. IEEE*, pp. 486–489, 2009.

[27] D. E. Pierre-Antoine, Thierry, "On the use of sparse time-relative auditory codes for music," *Department of Computer Science*.

[28] A. Holzapfel and Y. Stylianou, "Musical genre classification using nonnegative matrix factorization-based features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 424–434, feb 2008.

[29] Y. S. A. Holzapfel, "A statistical approach to musical genre classification using non-negative matrix factorization," *Acoustics Speech and Signal Processing 2007*, pp. 11–693.

[30] R. P. P. R. Panda, "Mirex 2012: Mood classification tasks submission," *Machine Learning*, vol. 53, pp. 23–69.

[31] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen, "A regression approach to music emotion recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 448–457, feb 2008.

[32] E. N. Tamatjita and A. W. Mahastama, "Comparison of music genre classification using nearest centroid classifier and k-nearest neighbours," in *2016 International Conference on Information Management and Technology (ICIMTech)*. IEEE, nov 2016.

[33] S. Saju, R. Rajan, and A. R. Jayan, "Music genre classification using spatan 6 FPGA and TMS320c6713 DSK," in *2017 International Conference on Signal Processing and Communication (ICSPC)*. IEEE, jul 2017.

[34] MARSYAS. (2018) Gtzan genre collection. [Online]. Available: www.marsyasweb.appspot.com/download/data_sets/

[35] C. Silla. (2008) Latin music database. [Online]. Available: https://sites.google.com/site/carlossillajr/resources/the-latin-music-database-lmd

[36] A. K. Carlos Silla Jr., Celso Kaestner, "Automatic music genre classification using ensemble of classifiers," *IEEE International Conference on Systems, Man and Cybernetics (SMC)*.

[37] A. Brainz. (2018) Pre-trained models. [Online]. Available: http://acousticbrainz.org/datasets/accuracy

[38] A. J. Izenman, *Modern Multivariate Statistical Techniques*. Springer New York, 2008.

[39] N. Bambrick, "Support vector machines: A simple explanation," 2018. [Online]. Available: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

[40] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[41] T. G. Dietterich, "Ensemble methods in machine learning," *International Workshop on Multiple Classifier Systems*, 2000.

[42] J. François, Y. Grandvalet, T. Denceux, and J.-M. Roger, *Bagging Improves Uncertainty Representation in Evidential Pattern Classification*. Heidelberg: Physica-Verlag HD, 2002, pp. 295–308. [Online]. Available: https://doi.org/10.1007/978-3-7908-1797-3_23

[43] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55–63, jan 1968.

[44] K. J. Cherkauer, "Human expert-level perfomance on a scientific image analysis task by a system using combined artificial neural networks," *Departament of Computer Sciences*, 1996.

[45] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on pattern analysis and machine intelligence*, 1998.

[46] G. B. Thomas G. Dietterich, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research 2*, 1995.

[47] T. M. Mitchell, *Machine Learning*. PN, 1990.

[48] J. M. Marin, *Análisis Discriminante*, UC3M, Ed., 2018. [Online]. Available: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/AMult/tema6am.pdf

[49] M. Mohammadi, F. Al-Azab, B. Raahemi, G. Richards, N. Jaworska, D. Smith, S. de la Salle, P. Blier, and V. Knott, "Data mining EEG signals in depression for their diagnostic value," *BMC Medical Informatics and Decision Making*, vol. 15, no. 1, dec 2015.

[50] A. J. Izenman, "Linear discriminant analysis," in *Springer Texts in Statistics*. Springer New York, 2013, pp. 237–280.

[51] Matlab. (2018) Matlab documentation, fitcdiscr. [Online]. Available: https://es.mathworks.com/help/stats/fitcdiscr.html

[52] B. W. Cheng Li. (2014) Fisher linear discriminant analysis. [Online]. Available: https://pdfs.semanticscholar.org/1ab8/ea71fbef3b55b69e142897fadf43b3269463.pdf

[53] C. A. de Armiño, M. Á. Manzanedo, and Á. Herrero, "Studying road transportation demand in the spanish industrial sector through k-means clustering," in *Advances in Intelligent Systems and Computing*. Springer International Publishing, jun 2018, pp. 387–396.

[54] U. Kużelewska, "Collaborative filtering recommender systems based on k-means multi-clustering," in *Contemporary Complex Systems and Their Dependability*. Springer International Publishing, may 2018, pp. 316–325.

[55] C. Marques, A. Mohsin, and J. Lengler, "A multinational comparative study highlighting students' travel motivations and touristic trends," *Journal of Destination Marketing & Management*, vol. 10, pp. 87–100, dec 2018.

[56] J. Wu, *Cluster Analysis and K-means Clustering: An Introduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–16.

[57] D. Olzewski, "K-means clustering of asymmetric data."

[58] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*. Hodder Education Publishers, 2001.

[59] V. K. Michael Steinbach, George Karypis, *A comparison of Document Clustering Techniques*, 2000, pp. 525–526.

[60] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 89–98.

[61] L. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, jan 1967.

[62] I. blogs, "Clusters spherical data," 2018. [Online]. Available: https://www.imperva.com/blog/2017/07/clustering-and-dimensionality-reduction-understanding-the-magic-behind-machine-learning/

[63] G. T. Fechner, *Elemente der Psychophysik (1860) (Classics in Psychology, 1855-1914) (Vols 4 & 5)*. Thoemmes Continuum, 1998.

[64] N. Degara, E. A. Rua, A. Pena, S. Torres-Guijarro, M. E. P. Davies, and M. D. Plumbley, "Reliability-informed beat tracking of musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 290–301, jan 2012.

[65] H. M. Ozaktas and M. A. Kutay, "The fractional fourier transform," in *2001 European Control Conference (ECC)*. IEEE, sep 2001.

[66] Wikipedia, "Signal filter using frfft," 2018. [Online]. Available: https://en.wikipedia.org/wiki/Fractional_Fourier_transform#CITEREFCandanKutayOzaktas2000

[67] alaiacanao. (2012) Fractional fft implementation in python. [Online]. Available: https://github.com/alaiacano/frfft

[68] M. T. Group. (2018) Essentia library. [Online]. Available: https://github.com/MTG/essentia

[69] Openpyxl. [Online]. Available: https://openpyxl.readthedocs.io/en/stable/#

[70] E. V. J. Salas, A. Torrente. (2018). [Online]. Available: http://ocw.uc3m.es/matematicas/algebra-lineal/teoria/algebra_teoria_14.pdf

[71] L. van der Maaten. (2008) t-sne mainpage. [Online]. Available: https://lvdmaaten.github.io/tsne/

[72] G. H. Laurens van der Maaten, "Visualizing data using t-sne," *Journal of Machine Learning Research*, pp. 2579–2605, 2008.

[73] C. McComb. (2015) Adjusted rand index. [Online]. Available: https://es.mathworks.com/matlabcentral/fileexchange/49908-adjusted-rand-index

[74] P. R. Kaufman L., "Finding groups in data: An introduction to cluster analysis," *NJ: John Wiley & Sons, Inc.*, 1990.

[75] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, dec 1985.

[76] D. Olszewski, "Asymmetric k -means clustering of the asymmetric self-organizing map," *Neural Processing Letters*, vol. 43, no. 1, pp. 231–253, mar 2015.

[77] B. S. Dominik Olszewski, "Asymmetric clustering using the alpha–beta divergence," *Pattern Recognition 47*, 2014.