UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

IMPLEMENTACIÓ D'UN SISTEMA NO SUPERVISAT AMB ADAPTACIÓ ONLINE PER CLASSIFICAR MÚSICA

Chan yong Yoon Kim

Director: Dr. Sergio Escalera Guerrero
Realitzat a: Departament de
Matemàtiques i Informàtica
Barcelona, 21 de juny de 2020

Resumen

A día de hoy, la música sigue siendo una parte importante de nuestras vidas. Actualmente, hay muchísimos estilos diferentes y para poder mantener un orden, se ha vuelto muy importante el análisis musical, ya sea para tenerlas clasificadas según el género o poder ser capaces de recomendar a los oyentes canciones similares a las que suelen escuchar.

En este proyecto, proponemos un sistema no supervisado capaz de clasificar canciones en diferentes estilos y, cuando es necesario, capaz de clasificar nuevos estilos que vayan apareciendo. Para conseguir esto, crearemos unos grupos a partir del algoritmo clustering jerárquico y, finalmente, clasificaremos y crearemos grupos nuevos a partir de las estrategias ECOC (Códigos de Corrección de Errores).

Resum

Avui en dia, la música segueix sent una part important de les nostres vides. Actualment, hi ha moltíssims estils diferents i per poder mantenir un ordre, s'ha tornat molt important l'anàlisi musical, ya sigui per tenir-les classificades segons el seu gènere o poder ser capaços de recomanar als oients cançons similars a les que solen escoltar.

En aquest projecte, nosaltres proposem un sistema no supervisat capaç de classificar cançons en diferents estils i, quan és necessari, capaç de classificar nous estils que vagin apareixent. Per aconseguir això, crearem uns grups a partir de l'algorisme clustering jeràrquic i, finalment, classificarem i crearem grups nous a partir de les estratègies ECOC (Codis de Correcció d'Errors).

Abstract

Nowadays, music is still an important part of our lives: every one of us listens to some kind of music. There are currently loads of different styles and, to keep them organized, musical analysis has become very important - either to classify them according to their genre or for being able to recommend listeners similar songs to the ones they usually listen to.

In this project we propose a unsupervised system to classify songs into different genres and, when necessary, label new genres that may come up. To achieve that, we will create groups from the Hierarchical clustering algorithm and, finally, classify and create new groups from the ECOC (Error-Correcting Output Codes) strategies.

Agradecimientos

Primero de todo, agradecer a mi tutor Dr. Sergio Escalera, ya que si no llega a ser por sus ideas y explicaciones de cómo manejar el proyecto y la paciencia que ha tenido conmigo yo no hubiese sido capaz de terminar este proyecto. Además, ha sabido cómo animarme para que no volviese a abandonar el proyecto.

Segundo, agradecer a mis compañeros de la carrera, que algunos ya son amigos para toda la vida, como Joaquim Comes, que me ha ayudado con la organización de los datos utilizados y, además, me proporcionó la idea de este proyecto. No todo ha sido siempre trabajar, también quiero agradecerles por todos esos momentos divertidos durante la carrera.

Por último, y no menos importante, agradecer a toda mi familia y amigos, en especial a mi hermano y a mi pareja, ya que sin su apoyo yo no habría llegado a donde estoy.

De corazón, muchísimas gracias a todos por haber sido tan buenos y pacientes conmigo, tengo la suerte de estar rodeado de gente maravillosa.

Índice

1 Introducción

La música ha existido desde los inicios del ser humano. Con el paso del tiempo, han ido cambiando sus instrumentos, sus representaciones, el estilo… Pero en cualquier momento de la historia, la música siempre ha sido esencial para la humanidad.

Gracias a la era digital en la que vivimos actualmente, podemos escuchar música casi en todas partes y casi en todo momento, lo que nos permite disfrutar de canciones que realmente se adaptan a nuestro gusto, incluso podemos encontrar música que no esperamos que nos guste.

La cantidad de música que nos rodea hoy es mucho mayor que en cualquier otro momento de la historia, lo que hace que sea realmente difícil encontrar canciones similares a lo que nos gusta. Debido a eso, nos encontramos que la clasificación de la música se ha vuelto muy necesaria, como por ejemplo, para poder organizarla y así escoger canciones similares o incluso poder encontrar algo nuevo, no sólo para las personas que desean encontrar el próximo éxito, sino también para las empresas que se dedican a la transmisión de música o para su distribución. Dada la gran cantidad de datos que nos encontramos, mi objetivo es hacer una prueba de concepto para ver si, con un conjunto de datos reducido, podemos llegar a organizarlos de alguna manera.

Hoy en día, nos encontramos que ya existen muchos algoritmos para extraer características de audio que van desde frecuencia, tono, timbre, ritmo, etc. Esto nos servirá para reducir la cantidad de datos que contiene una canción. Muchos de los estudios encontrados suelen utilizar MFCC, que apareció como una función para el procesamiento de voz dada que la señal era demasiado compleja para analizarla usando técnicas de procesamiento de señal más simples.

Utilizando MFCC, acabaremos teniendo los datos más simplificados, pero con suficiente información restante para luego poder usarlo con enfoques de aprendizaje automático [1]. Algunos de los algoritmos más utilizados actualmente son AdaBoost y Support Vector Machine en caso de aprendizaje supervisado y K-means para aprendizaje no supervisado.

Con la aparición del Deep Learning, la mayoría de los estudios en el campo utilizan este enfoque, debido a su capacidad de determinar la precisión de su propio modelo y adaptarse a sus resultados. El problema es que, con un conjunto de datos reducido en un problema no supervisado como el nuestro, los métodos más clásicos (de aprendizaje automático) se usan con mayor frecuencia. La razón principal es que los modelos de Deep Learning involucran millones de parámetros para ser entrenados, lo que requiere de una gran base de datos. Se pueden encontrar ejemplos de estrategias de Deep Learning que utilizan grandes cantidades de audios para los interesados en los estudios de Miguel Flores [2] y en el proyecto DeepSound [3].

Teniendo en cuenta lo anterior, mi objetivo personal es aprender a procesar datos digitales de audio y, sobretodo, machine learning para ver las posibilidades de aprendizaje online/adaptativo que permiten hacer. Entonces, nuestra propuesta es desarrollar un

algoritmo de aprendizaje no supervisado que clasifique, por su similitud, un conjunto de datos de audios relativamente pequeño en grupos y si aparece un elemento que no se parece a ningún otro crear un nuevo grupo.

Para representar nuestra base de datos, utilizaremos las técnicas que se han encontrado más confiables cuando se trata de representar datos de audio, MFCC, que extraerá las características más importantes de cada pista y nos ayudará a evitar trabajar con todos los datos sin procesar.

Después, utilizaremos el algoritmo de clustering jerárquico que sirve para tratar un número indeterminado de datos en un escenario no supervisado. Con esto, queremos conseguir agrupar nuestros datos en grupos diferentes, sin saber a qué género pertenece cada canción, y, viendo su similitud, ser capaces de clasificar audios y también ser capaces de crear nuevos grupos en caso de que el audio no se parezca a ninguno de los creados anteriormente.

Finalmente, veremos que clustering jerárquico nos traerá un problema para hacer la parte de aprendizaje online/adaptativo (crear grupos nuevos) y, para solucionarlo, utilizaremos los Códigos Correctores de Errores (Error-Correcting Output Codes). ECOC nos permitirá clasificar los audios en los grupos creados con anterioridad y permitirá realizar un aprendizaje online/adaptativo de forma mucho más eficiente.

El resto del proyecto lo organizamos como sigue, después de realizar el algoritmo de aprendizaje no supervisado, queda evaluar los resultados obtenidos y, finalmente, acabaríamos con las conclusiones.

Este proyecto utilizará pistas de audio públicas y comparará diferentes características y relevancia para proporcionar un sistema automático que funcione completamente, que en el futuro podría usarse en proyectos más grandes, como para un recomendador.

2 Datos y extractor de características

Para llevar a cabo nuestro proyecto, nos centramos en una base de datos pública y todas las técnicas de análisis de datos que detallaremos precisan de un extractor de características. Para conseguir estas características de audio, utilizaremos MFCC y, finalmente, PCA para reducir el tamaño de la matriz dada por MFCC. Como ya hemos comentado al inicio de la memoria, sabemos que existen métodos más óptimos para la extracción de las características de audio, como por ejemplo con Deep Learning, nosotros fijamos un vector de características muy estándar, no el mejor, para comprobar cómo de bien van las estrategias online.

2.1 Conjunto de datos

Los datos los encontramos en una página web, donde ya venían los audios preparados[4]. Para nuestros análisis hemos utilizado un conjunto de 1000 canciones de 10 géneros diferentes. Los géneros no son del todo necesarios ya que el objetivo es crear un sistema de aprendizaje no supervisado, pero los aprovechamos para tener una idea de qué subconjunto utilizar para, más tarde, experimentar. Los géneros que contiene son Rock, Country, Metal, Reggae, Pop, Clásica, Disco, Jazz, Blues y Hip Hop. Todas las canciones tienen una duración de 30 segundos y todas contienen propiedades idénticas. Todos los datos están disponibles para cualquier persona y se puede utilizar para investigación sin ningún tipo de cargo.

Para trabajar con los audios es importante conocer algunos conceptos básicos, como no todo el mundo tiene conocimiento de estos conceptos, explicaré brevemente los que necesitamos saber:

- Fotograma de audio: información de un margen de tiempo dado.
- Frecuencia de muestreo: número de muestras tomadas de una señal continua para producir una señal discreta.
- Canales: número de transmisiones que se envía el audio.
- Tamaño del fotograma: tamaño de cada fotograma. Frecuencia de muestreo * canales.
- Marcos por segundo: número de fotogramas por segundo. Tamaño del marco / s.

Todas las canciones tienen una duración de 30 segundos y contienen unas propiedades similares:

- Frecuencia de muestreo: 22050Hz
- Canales: 1 (Mono)
- Marcos por segundo: 22050 fps (del inglés Frames per second)

2.2 Vector de características

Como ya comentamos anteriormente, nosotros utilizaremos el MFCC para la extracción de las características de audio. Los MFCC (Coeficientes Cepstrales de las Frecuencias de Mel,

del Inglés, Mel Frequency Cepstral Coefficients) se usan generalmente para extraer características de la voz, pero hoy por hoy se han utilizado en diversos experimentos para todo tipo de sonidos. Este método fue creado por Paul Mermelstein y S. Davis.

Utilizando este método, terminaremos con una matriz cuyo tamaño dependerá de la cantidad de coeficientes que queramos y la duración del audio. Ahora explicaremos brevemente cada paso del algoritmo[5][6]:

1. Dividir la señal en intervalos del mismo tamaño.
   El primer paso se toma el audio y se separa en n-trozos del mismo tamaño. El tamaño dependerá de las características del archivo, pero normalmente se usa un tamaño entre 20 y 30 ms.

2. Aplicar la Transformada de Fourier en cada intervalo.
   En el segundo paso, se le aplicará la Transformada de Fourier para obtener las frecuencias de cada intervalo y se tendrá un conjunto de finito componentes que formarán la señal original.

3. Convertir los valores a Mel Scale (Escala de Mel).
   En este paso se conseguirá obtener el valor correspondiente de cada frecuencia en su escala de Mel. Esta escala representa tonos, que para los oyentes, son parecidos. En la Fig2.1 podemos ver la Escala de Mel.

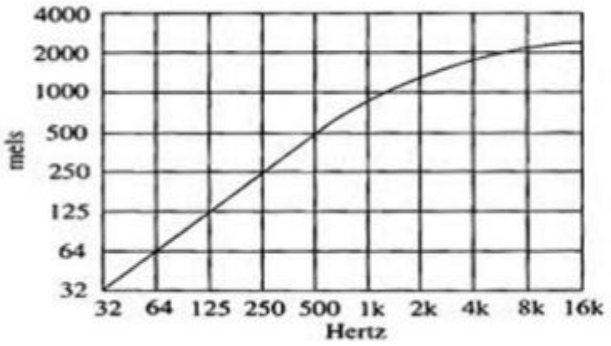


Fig2.1. Escala de Mel

4. Coger los valores de cada Frecuencia de Mel.
   Una vez convertidos los valores en la Escala de Mel se guardarán.

5. Aplicar la transformada de coseno directa a estos valores.
   Este último paso es para volver a convertir los valores de la Escala de Mel en la escala original. Los valores resultantes serán los MFCC.

Ahora, que ya hemos extraído las características, tenemos que decidir qué hacer con ellas, cómo representarlas, ya que la cantidad de características obtenidas es mucho mayor que el conjunto de datos con el que trabajamos. En nuestro caso, hemos decidido representar estas características creando histogramas de cada componente.

La idea es reducir la cantidad de valores que tenemos manteniendo toda la información posible. Para conseguir esto, tendremos tantos histogramas como coeficientes hemos utilizado y, finalmente, tendremos que cada canción estará representada por una matriz cuyo tamaño dependerá de la cantidad de coeficientes y la cantidad de pasos(Steps) que deseemos. Los histograma se construyen de la siguiente manera:

1. Se obtienen los máximos y mínimos de todos los conjuntos.
2. Se divide el intervalo en tantos pasos(Steps) como se prefiera.
3. Se crea un histograma para cada coeficiente.
4. Se pone cada valor correspondiente en su intervalo.
5. Se divide cada valor final por la cantidad de valores que se tengan.
6. Se concatena cada histograma en una matriz unidimensional.

Ahora que ya hemos obtenido la representación del vector de características, realizaremos una última modificación mediante PCA (Análisis de componentes principales, del Inglés Principal Component Analysis) para reducir todavía más el tamaño de la matriz.

PCA realiza transformaciones ortogonales de un conjunto que puede tener correlación y que no tenga correlación lineal. El algoritmo se realiza de la siguiente manera[7][8]:

1. Normalizar.
   Primero se normalizan todos los datos para que sea más fácil de calcular el siguiente paso.

2. Calcular la matriz de covarianza.
   Siguiente paso, crear una matriz con la covarianza de cada una de las características.

3. Encontrar los vectores y valores propios de la matriz.
   Con este paso, se tendrán tantos vectores propios como sea necesario para el tamaño de nuestros datos.

4. Reorganizar los datos.
   Ahora se multiplica la matriz original por los vectores propios. Con esto se consigue que la matriz original se convierta en una matriz de dimensión más pequeña.

Con todo lo explicado en este apartado, ya tendríamos preparado nuestro conjunto de datos para poder experimentar con ellos.

3 Aprendizaje no supervisado

Nosotros queremos un sistema no supervisado online, es decir, desconocemos qué tipos de audios tenemos inicialmente y los queremos agrupar según su parecido. Después, cuando nos vengan canciones nuevas, clasificarlas en algún grupo de los creados anteriormente y, si hay alguna canción que difiere mucho de los grupos existentes, crear un grupo nuevo automáticamente (online).

Después de analizar sobre los diferentes algoritmos de aprendizaje no supervisado, nos decidimos por el de clustering jerárquico porque no nos hacía falta definir, si no queremos, el número de agrupaciones. Sabemos que actualmente hay muchísimos algoritmos diferentes que agrupan igual, o mejor, que el clustering jerárquico de manera no supervisada, pero encontrar el mejor no es nuestro objetivo, simplemente vimos que, en nuestro caso, con la agrupación jerárquica las canciones se repartían bien por los diferentes grupos y lo consideramos suficiente[9][10].

3.1 Clustering jerárquico

Como hemos comentado, para nuestro sistema, hemos escogimos el clustering jerárquico aglomerativo, es decir, se empieza asignando a cada canción un cluster y luego se van uniendo según parecido hasta que queden todos agrupados en un único cluster o hasta que se hayan creado el número de clusters que hayamos definido. El algoritmo funciona de la siguiente manera[11][12]:

1. Cada elemento se considerará un cluster.
   Para empezar, todas las canciones serán un cluster diferente.

2. Calcular por cada cluster la distancia con los demás clusters.
   Para este paso, ya se debería de haber decidido qué método se utiliza para calcular la distancia entre clusters y el linkage(enlace), es decir, la manera para decidir qué dos clusters se parecen. En la Fig3.1 podemos ver un ejemplo de distancias entre clusters.

	A	B	C	D	E	F	G
A	0						
B	2,15	0					
C	0,7	1,53	0				
D	1,07	1,14	0,43	0			
E	0,85	1,38	0,21	0,29	0		
F	1,16	1,01	0,55	0,22	0,41	0	
G	1,56	2,83	1,86	2,04	2,02	2,05	0

Fig3.1. Distancias de cada cluster con el resto[13].

3. Encontrar los dos clusters más similares según linkage y juntarlos.
   Una vez se hayan calculado las distancias y según el linkage que se haya escogido, se decidirá qué clusters juntar. Tomando de ejemplo las distancias de la Fig3.1 si utilizamos un linkage de tipo Single(o Mínimo), juntaremos los clusters C y E.

4. Continuar este proceso hasta tener un único cluster grande o hasta tener el número de clusters definido.

Acabamos de explicar la manera de crear las diferentes agrupaciones. Finalmente, solamente quedaría clasificar las nuevas canciones que vayan entrando y sería calcular la distancia entre esta nueva canción con cada cluster y añadirla al más parecido, pero si viéramos que para todos los clusters hay una diferencia grande, entonces indicaremos que se forme un cluster nuevo.

4 Error Correcting Output Codes para aprendizaje online

Con clustering jerárquico, podríamos decir que ya se ha cumplido el objetivo del proyecto, conseguir agrupar sin tener información previa de los datos (aprendizaje no supervisado), clasificar los nuevos datos que vayan entrando y si son muy diferentes crear un nuevo grupo (aprendizaje online). Pero, surge un problema, cada vez que aparezca una canción nueva que no tenga parecido con el resto de clusters y, por lo tanto, queramos añadir un cluster nuevo, se tiene que recalcular toda la estructura añadiendo la nueva canción a la base de datos. Dado que uno de los objetivos es que nuestro sistema con el tiempo vaya aprendiendo información nueva, clases nuevas, y el hecho de tener que volver a recalcular siempre lo aprendido previamente añadiendo la nueva información, llegará un momento que habrán tantos datos para aprender que el sistema ya no será tan óptimo.

Para evitar este problema, se utilizará la estrategia ECOC, Códigos Correctores de Errores (del Inglés, Error Correcting Output Codes). ECOC es una estrategia de aprendizaje por ensamblado, es decir, combina múltiples algoritmos de aprendizaje para obtener un mejor rendimiento a la hora de clasificar los datos y también nos permitirá incluir o quitar aprendizaje previo sin tener que re-entrenar toda la estructura, lo que nos permite realizar el aprendizaje online sin entrenar todo el sistema cada vez que aparezca un tipo de dato nuevo[14]. ECOC es una estrategia sencilla, y poderosa, para hacer frente a los problemas de multiclase a partir de la combinación de clasificadores binarios.

En nuestro trabajo, utilizamos esta estrategia por su gran eficiencia en el ámbito de aprendizaje online y consideramos que cualquier clasificador base puede ser usado independientemente de la estrategia ECOC online. Para nuestros experimentos, y a modo de ejemplo, utilizaremos un SVM (Support Vector Machine) lineal. Como trabajo para el futuro, sería interesante incluir modelos de Deep Learning a esta estrategia de ensamblado.

4.1 ECOC

Dado un conjunto de N clases, la técnica ECOC diseña una palabra, un código, para cada clase. Estos códigos se almacenan en una matriz $M \varepsilon \{-1, 1\}^{N \times n}$, siendo n la longitud del código. La matriz M se construye considerando n problemas binarios (n clasificadores binarios), cada uno corresponderá a una columna de la matriz. Según la estrategia de codificación binaria escogida, cada columna tendrá -1, 1 dependiendo del problema binario o 0 si la clase no está considerada en ese problema binario. En la Fig4.1 podemos ver un ejemplo de cómo sería una matriz ECOC.

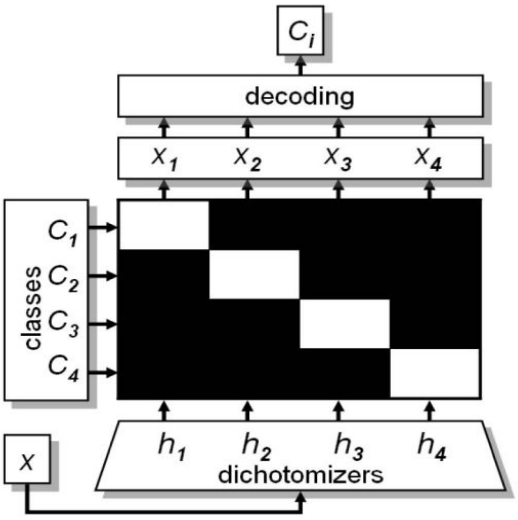
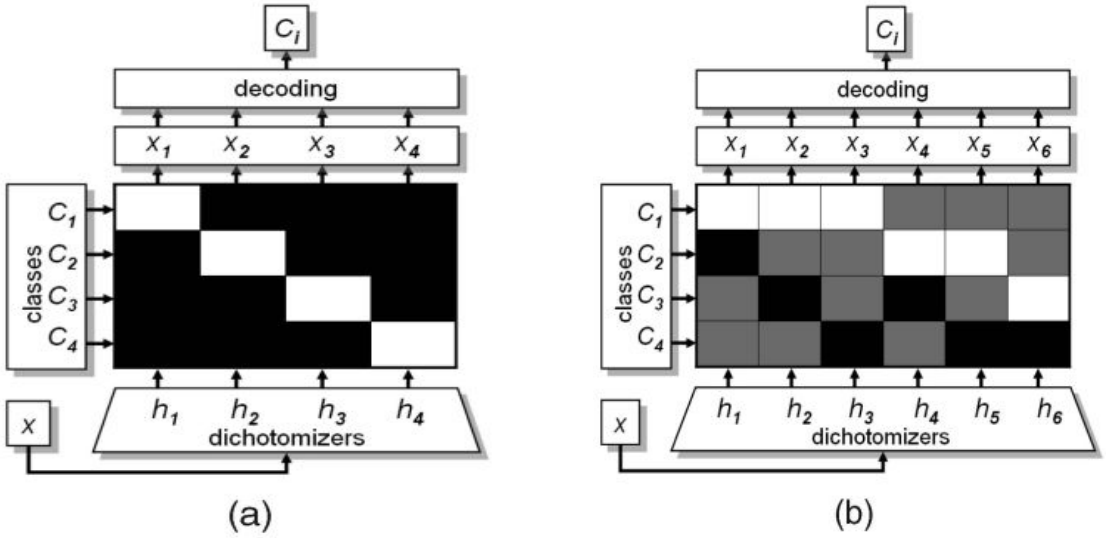


Fig4.1. Ejemplo estructura matriz ECOC. Los blancos son 1 y los negros son los -1.

Las estrategias de codificación binaria más conocidas son las de uno contra todos, donde cada clase se distingue de todo el resto o, dicho de otra manera, cada clasificador binario (o dicotomía) está entrenado para distinguir una clase del resto de clases, y la estrategia densa aleatoria, donde se genera una matriz aleatoria M maximizando la separabilidad de filas y columnas en términos de la distancia de Hamming. Para estas dos estrategias, solamente se utilizaban los valores -1 y 1. Más adelante, Allwein introdujo un tercer símbolo, el valor 0, y este nos permite que algunas clases sean ignoradas. Entonces, la matriz de codificación pasa a permitir un valor más $M \varepsilon \{-1, 0, 1\}^{N \times n}$, donde el 0 indica que una clase en particular no se considera en el clasificador binario. Gracias a esto, surgió la estrategia uno contra uno[15]. En la Fig4.2 podemos ver como quedan las matrices de codificación según estrategia, las posiciones blancas representan el valor 1, las posiciones negras representan -1 y las grises representan el 0.



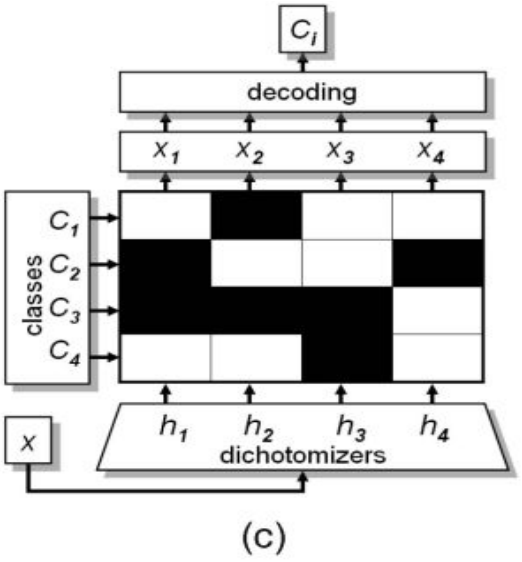


Fig4.2. (a) uno contra todos, (b) una contra uno, (c) densa aleatoria.

Una vez entrenados los clasificadores binarios y creado la matriz de codificación, ahora nos toca conocer la decodificación la cual nos servirá para clasificar los elementos nuevos que nos vayan entrando. Entonces, aplicando los n clasificadores a cada elemento, obtendremos una palabra código. Estos códigos de cada elemento se comparan con cada fila de la matriz M (recordamos que cada fila contiene una palabra código de cada clase) y al elemento se le asigna la clase con la que tenga la distancia más pequeña. Los métodos de decodificación más utilizados son la Decodificación Euclidiana y la Decodificación de Hamming[15], por otro lado, para la decodificación ternaria, es decir, los casos en los que incluímos el valor 0, como la estrategia uno contra uno y, como veremos más adelante, para el ECOC online, tenemos la decodificación Loss-Weighted.

1. Decodificación Euclidiana: se basa en calcular mediante la distancia Euclidiana.

2. Decodificación de Hamming: se basa en calcular las distancias mediante la distancia de Hamming, este método indica cuántos bits son diferentes.

3. Decodificación Loss-Weighted, es una variante para no tener en cuenta las predicciones de los clasificadores binarios donde en la matriz M tienen el valor 0, es decir, por cada vez que calculamos una distancia hay que multiplicar por el valor de la matriz con el que estamos calculando. La ecuación para el cálculo de una predicción P con una fila i de la matriz M sería la siguiente:

$$d(P,\ M_i)\ =\ \sum_{0}^{n} |M_{i\,n}|\ \cdot\ FD(M_{i\,n},\ P(n))$$

donde P es una predicción dada por los n clasificadores binarios (contendrá n elementos), M_i es la fila i de la matriz M, $M_{i\,n}$ es el bit de la fila i columna n de la matriz M, FD es la función distancia que podría ser la distancia de hamming por ejemplo y P(n) es el bit n de la predicción P.

4.2 ECOC online

ECOC originalmente se definió para solucionar problemas multiclase, se utilizó para conseguir que clasificadores binarios pudieran distinguir N clases. Era una forma de extender los clasificadores binarios a clasificadores multiclase y además permiten corregir errores de los clasificadores.

Con el paso del tiempo, todas las técnicas van mejorando para solucionar nuevos problemas que van surgiendo, uno de los problemas que surgieron fue la inclusión de nuevas clases sin tener que re-entrenar toda la información previa aprendida. En 2011, por primera vez se adaptó el ensamblado ECOC para poderlo utilizar como método de aprendizaje online[14].

Para añadir una nueva clase a la matriz ECOC se han de realizar dos pasos, el primero se debe definir una palabra código para esta nueva clase, es decir, añadir una fila más a la matriz M. El segundo paso, se ha de aprender un nuevo clasificador binario que aprenda sobre esta nueva clase, dicho de otra manera, añadir una columna más a la matriz M. Por supuesto, siempre manteniendo el resto de datos de la matriz intacta. En la Fig4.3 podemos ver un ejemplo de cómo queda la matriz ecoc añadiendo una clase nueva donde las posiciones en blanco son los 1, los negros son -1 y los grises son 0.

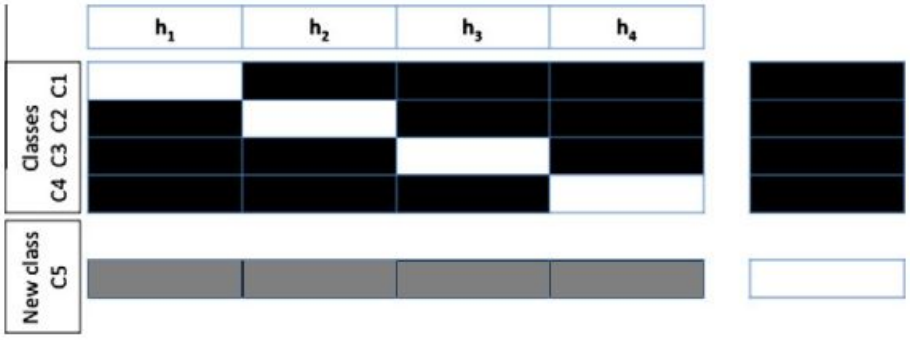


Fig4.3. Ejemplo estructura de la matriz ECOC uno contra todos añadiendo una nueva clase.

Siguiendo el ejemplo de la Fig5.3, vemos que la información previa aprendida sigue intacta y, para la nueva clase que se va a aprender, tenemos el código (0, 0, 0, 0, 1). Se colocan esos ceros porque no queremos que se tenga en cuenta la predicción de estos clasificadores ya que estos no fueron entrenados con la nueva clase.

Para la decodificación, como ahora se obtiene una matriz ternaria, es decir, con 3 posibles valores (-1, 0 y 1), tendremos que utilizar decodificación Loss-Weighted, pero con un añadido más a la fórmula, dividir por la cantidad de bits de la fila i diferentes de 0. La fórmula quedaría de la siguiente manera:

$$d(P,\ M_i) = \frac{\sum_{0}^{n} |M_{i\,n}| \cdot FD(M_{i\,n}, P(n))}{\sum_{0}^{n} |M_{i\,n}|}$$

donde P es una predicción dada por los n clasificadores binarios (contendrá n elementos), M_i es la fila i de la matriz M, M_{in} es el bit de la fila i columna n de la matriz M, FD es la función distancia que podría ser la distancia de hamming por ejemplo y P(n) es el bit n de la predicción P.

5 Evaluación

Hasta ahora hemos explicado nuestra base de datos, los métodos que utilizaremos para sacar la mayor información posible y también hemos hablado de las diferentes técnicas que utilizaremos para llevar a cabo nuestros experimentos. A partir de ahora, explicaremos cómo hemos realizado estos experimentos y los resultados obtenidos.

5.1 Detalles de implementación

Para llevar a cabo los experimentos, hemos realizado el proyecto en un sistema operativo Linux Mint 19.1 Cinnamon y en un lenguaje de programación bastante conocido como es Python en la versión 3.0.6, aprovechando sus múltiples librerías para poder realizar las diferentes técnicas comentadas anteriormente.

Las librerías que hemos utilizado son librosa[16] y sklearn[17]. Librosa contiene métodos para el análisis de audio y la librería sklearn contiene la mayoría de los algoritmos de aprendizaje automático que utilizaremos en los experimentos.

De librosa solamente utilizaremos las funciones:

- Load: esta función permite cargar los archivos de audio. Modificamos los parámetros de la función para que todos los archivos tengan las mismas propiedades, las que comentamos anteriormente en el apartado 3.1 Conjunto de datos. Los parámetros que necesitamos son:

    - duration: indicaremos con un float y sirve para recortar la canción y tener la duración que queramos. En nuestro caso, está fijado en los primeros 30 segundos.

    - mono: enviando un bool indicaremos si queremos que el audio sea monocanal, es decir, que tenga un único canal de transmisión. Nosotros enviamos valor True para que todas sean monocanal.

    - sr: indicamos con un integer y sirve para editar la frecuencia de muestreo. Nosotros lo fijamos en 22050Hz.

- Mfcc: función que calcula los MFCC de los audios que cargaremos previamente con la función load. Esta función se ajustará automáticamente para intentar devolver la matriz lo más pequeña posible sin perder grandes cantidades de información. De todos los parámetros, solamente nos interesa n_mfcc, que es la cantidad de mfcc que nos devolverá, el resto de parámetros sirven para modificar las propiedades del audio, cosa que ya los modificamos con la función load. Nosotros fijamos n_mfcc en 15.

De sklearn utilizaremos los siguientes algoritmos:

- PCA: esta función aplicará el algoritmo Principal Component Analysis, explicado anteriormente, y hará que se reduzcan la cantidad de valores que contiene el vector de características. El parámetro a tener en cuenta es n_components que es la cantidad de datos a la que reducirá el algoritmo a cada vector de características. Nosotros lo tenemos fijado en 100 componentes ya que es el 10% de características que se recomiendan.

- AgglomerativeClustering: este método aplicará el clustering jerárquico aglomerativo para crear los grupos de audios con los que empezaremos a trabajar. Los parámetros que utilizamos son:
    - n_clusters: se indica con un integer y sirve para indicar el número de clusters, grupos, que queremos conseguir. Nosotros lo tenemos fijado en 10.

    - affinity: se indica con un string y sirve para indicar qué métrica utilizar para calcular las distancias entre clusters. Hay bastantes tipos diferentes a utilizar, nosotros utilizaremos la distancia euclidiana.

    - linkage: se indica con un string y sirve para definir qué criterio se utilizará para unir dos clusters. Nosotros utilizamos el criterio ward, que minimizará la varianza de los clusters que se vayan uniendo.

Para la parte del ensamblado de ECOC, hemos escogido una codificación uno contra todos ya que cada cluster creado es diferente de los demás. Para la decodificación, utilizaremos la distancia de Hamming. Para la parte de la adaptación online, seguiremos trabajando con la codificación uno contra todos, ya que la gracia del ECOC online es aprender información nueva sin retocar lo aprendido anteriormente, pero para la parte de decodificación, utilizaremos la fórmula explicada anteriormente utilizando la distancia de Hamming. En los dos casos, para todos los clasificadores binarios, o dicotomías, utilizaremos, de la librería sklearn, la función LinearSVC que aplica el algoritmo Support Vector Machine lineal con todos los parámetros por defecto.

En general, para todas las funciones que cogemos de librerías, tienen unos parámetros muy genéricos, y eso es porque nuestro objetivo no es encontrar el mejor tipo de algoritmo que utilizamos para que nuestro sistema funcione mejor, si no ver que, con unos algoritmos muy estándar, podemos realizar un aprendizaje no supervisado con adaptaciones online sin tener que re-entrenar lo aprendido previamente.

5.2 Procedimiento de evaluación

En nuestra base de datos tenemos 10 géneros y 100 canciones por géneros, es decir, un total de 1000 canciones, conocemos a qué género pertenece cada una, pero como ya comentamos anteriormente, es algo que nosotros sabemos pero para el sistema será desconocido.

Antes de empezar los experimentos, habremos realizado la extracción de los mfcc (vector de características) de cada audio de la base de datos y todos ellos los iremos añadiendo en una matriz. Después, como la cantidad de características es mucho mayor que el conjunto de datos, intentaremos reducir los datos mediante histogramas, método explicado en los apartados anteriores, y, finalmente, aplicaremos el algoritmo PCA para reducir, todavía más, la cantidad de datos. Una vez preparado el conjunto de datos, ya podemos empezar a hablar de los experimentos.

Para poder entrenar nuestro sistema y luego poder validarlo, separaremos las canciones de metal y clásica del resto y, además, crearemos dos conjuntos a partir de las 800 canciones restantes, uno será el conjunto de train, que tendrá el 95%, para entrenar el sistema y el otro conjunto será el de test, que se llevará el 5%, que servirá para validar el sistema aprendido junto con los dos géneros que separamos.

Para validar la primera parte, clustering jerárquico, entrenaremos el sistema con el conjunto de train y calcularemos los centroides de cada cluster generado. Después, calcularemos las distancias mínimas, con distancia euclidiana, entre el conjunto de test y los centroides y guardaremos esas distancias. Luego, haremos lo mismo, pero con todas las canciones de metal y clásica, en principio, como estos dos géneros no se entrenaron con el resto, deberían de dar una distancia media más grande que el conjunto de test. Después de verificar si las distancias han salido como esperábamos, re-entrenaremos el sistema añadiendo metal y clásica al conjunto de train y de test, 95% y 5% respectivamente, y volveremos a calcular distancias mínimas entre el conjunto de test y los nuevos centroides. Ahora que el sistema si está entrenado con metal y clásica, la distancia media debería de haber bajado.

Para la validación de la segunda parte, estrategia ECOC, haremos los mismos experimentos pero con el sistema ECOC, utilizando los clusters ya generados por el clustering jerárquico, una codificación uno contra todos y la decodificación de Hamming explicado anteriormente, y verificaremos que con esta estrategia llegamos a la misma conclusión que clustering jerárquico antes de ser re-entrenado con metal y clásica. Finalmente, si la hipótesis se cumple, pasaremos a realizar el ECOC online y verificaremos que, cuando añadamos la clase metal, la distancia se hará más pequeña, pero con clásica seguirá siendo grande y, por último, añadir clásica como nueva clase y ver que la distancia vuelve a disminuir.

Para saber si los resultados son satisfactorios, tendremos que ir comparando las distancias calculadas en cada experimento y verificar que cada vez que realizamos un cambio la media de la distancia mínima cada vez es menor.

5.3 Evaluación de clustering no supervisado

Una vez listos los conjuntos de train y test, y teniendo por separado las canciones de metal y clásica podemos comenzar con el primer experimento, que lo separaremos en dos partes. Primera parte, creamos los 10 clusters, utilizando el conjunto de train, con el clustering

jerárquico. Luego, calculamos los centroides de cada cluster y realizamos la distancia euclidiana entre los 10 centroides y el conjunto de test y nos guardamos la distancia mínima de cada canción. En la Fig5.1 (a) podemos ver la distancia mínima de cada canción del conjunto test a los centroides. Después, realizamos el mismo cálculo de las distancias mínimas, pero con todas las canciones de metal y clásica, en vez de con el conjunto de test. En la Fig5.1 (b), podemos ver estas distancias calculadas. En el apéndice 1, mostramos cómo han quedado distribuidos los audios del conjunto de train por los diferentes clusters generados. En el apéndice 2, mostramos dónde se clasificarían las canciones de metal y clásica con las distancias lejanas obtenidas.

Para la segunda parte, volveremos a calcular los clusters, pero añadiendo el 95% de metal y clásica al conjunto de train y el 5% a conjunto de test y, finalmente, volveremos a calcular las distancias euclidianas entre el conjunto test y los nuevos centroides. En la Fig5.1 (c) podemos ver las distancias mínimas de cada canción de este nuevo conjunto test con los nuevos centroides. En la Tabla5.1 mostramos las medias de las distancias obtenidas.

```
[0.8057 0.8115 0.7918 0.619  0.6483 0.7308 0.6443 0.9611 0.6252 0.6614
 0.5578 0.6464 0.5949 0.7005 0.2844 0.6183 0.4267 0.5709 0.4042 0.5739
 0.81   0.6088 0.5681 0.6396 0.6836 0.5417 0.931  0.8917 0.8475 0.8352
 1.1497 0.8881 0.9456 0.7869 0.6266 0.5602 0.7889 0.401  0.8258 0.6496]
```

(a)

```
[0.7581 1.0122 1.0719 0.9207 1.2549 0.9551 1.1904 0.9356 1.1088 0.9005
 0.6049 1.3489 1.1183 0.8451 0.8697 1.0236 0.7869 0.8016 0.7185 0.7842
 0.715  1.0038 0.8531 0.7826 0.7185 0.8844 0.8737 1.1302 0.9266 0.8851
 1.1032 0.7694 0.7844 0.8775 0.9424 0.9629 0.9744 1.0658 0.9291 0.6609
 1.3384 0.7927 0.6834 0.8242 1.2417 1.1511 1.4151 0.7359 0.8851 1.0122
 0.8475 0.8469 0.9046 0.7842 1.0517 1.2655 1.0138 0.9459 1.4644 0.772
 0.8668 0.8785 1.2373 0.9728 0.6611 1.2749 0.7377 0.9077 0.7359 0.5191
 0.7301 1.005  0.8737 1.1778 0.7146 1.0766 0.6178 1.2522 0.844  0.8837
 0.9107 1.0152 0.8448 1.0144 0.9162 1.3103 1.0432 0.8399 0.7146 0.9486
 0.8307 0.7185 1.0969 0.5776 1.0085 1.3507 0.7505 0.7672 0.6727 1.1033
 0.983  0.8052 1.2063 0.9506 0.6975 0.8494 1.0443 0.939  0.6981 0.7195
 0.8144 0.7084 0.912  1.0549 0.9337 0.9803 1.24   1.1967 0.9175 1.0495
 0.8127 0.8302 0.713  1.0114 0.7533 0.7092 0.7382 0.6129 0.6952 0.8956
 0.5271 0.6354 1.0863 0.799  0.7239 0.7003 1.2917 0.7858 0.9067 0.9165
 1.4649 1.018  0.7112 1.1035 0.7558 0.8876 0.8891 0.5273 0.6999 0.6753
 0.7257 0.8521 1.1169 0.8843 0.5978 1.0476 0.6517 0.9591 0.7943 1.2101
 0.6301 0.8178 1.0513 1.0125 0.5943 0.8101 1.0941 0.7549 0.9048 0.6848
 1.144  0.6692 0.6033 0.8569 0.7945 0.7468 0.8332 1.0166 0.6544 0.7611
 0.7591 0.9935 0.7324 0.9894 0.6427 1.1147 0.6268 0.7978 0.7595 0.6902
 0.7147 1.0464 0.8401 0.8931 0.858  0.6514 1.2442 0.5882 1.0556 0.7167]
```

(b)

```
[0.8464 0.8519 0.7253 0.7017 0.6554 0.7324 0.5162 1.0654 0.6967 0.7953
 0.7034 0.6203 1.352  0.9793 0.9524 0.5302 0.6867 0.6179 0.7164 0.3598
 0.6261 0.4155 0.5543 0.4408 0.5012 0.7121 0.8575 1.5762 0.6708 0.7635
 0.8039 0.6422 0.5262 0.6515 0.6765 0.6188 0.9521 0.7698 0.8988 0.8954
 1.1693 0.8883 0.9126 0.6979 1.1405 0.6303 0.7526 0.5211 0.8075 0.6588]
```

(c)

Fig5.1. **(a)** Las distancias mínimas de cada canción del conjunto test con los centroides. **(b)** Distancias mínimas de cada canción de metal y clásica con los centroides. **(c)** Las distancias mínimas de cada canción del nuevo test con los nuevos centroides.

Distancia realizada entre	Medias de las distancias mínimas
Conjunto test y centroides	0.6914
Todo metal y clásico y centroides	0.8940
Conjunto test completo y nuevos centroides	0.7467

Tabla5.1. Medias de las distancias mínimas obtenidas.

Como podemos observar en la Tabla5.1, antes de entrenar el sistema con metal y clásica, la distancia media entre test y centroides es menor que la distancia media entre metal-clásico y centroides, algo lógico ya que el sistema no conoce nada sobre metal y clásica. Después, cuando re-entrenamos el sistema añadiendo los datos de metal-clásica, vemos que las medias de la distancia del test antiguo y del nuevo test a sus respectivos centroides son muy similares.

Con este experimento, hemos podido comprobar que clustering jerárquico funciona correctamente y que podría ser una solución online si no tuviese el problema de que cada vez que queramos aprender alguna clase nueva, tenemos que re-calcular siempre todo con todos los datos, incluyendo los nuevos. Esto implica que en un futuro, poco a poco, los datos a re-entrenar serán tantos que el sistema dejará de ser eficiente. Por lo tanto, necesitamos un sistema escalable, es decir, un sistema que para aprender algo nuevo pueda mantener lo aprendido previamente y pueda adaptar esta información nueva, es decir, añadir información nueva sin tocar lo antiguo.

5.4 Evaluación de ECOC

Para este segundo experimento, utilizaremos los primeros 10 clusters generados anteriormente con el clustering jerárquico. Para la codificación de la matriz M, utilizamos la estrategia uno contra todos y los clasificadores binarios serán todos SVM lineal. Luego, para la decodificación, utilizando los datos de test y todo metal-clásica, emplearemos la distancia de Hamming para calcular las distancias mínimas. En la Fig5.2 podemos ver las distancias mínimas obtenidas. En la Tabla5.2 mostramos las medias de las distancias obtenidas.

```
[0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0.]
```

(a)

```
[1. 2. 1. 0. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1.
 1. 0. 1. 1. 0. 1. 0. 1. 0. 2. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 1. 2. 2. 0. 0. 1. 1. 2. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0.
 1. 0. 1. 0. 0. 2. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 2. 2.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0.
 1. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1.
 0. 1. 1. 1. 0. 1. 1. 2. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 0.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 1. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 1. 1. 1.]
```

(b)

Fig5.2. **(a)** Distancias mínimas de cada canción de test a las filas de la matriz M. **(b)** Distancias mínimas de cada canción de metal-clásica a las filas de la matriz M.

Distancia realizada entre	Media de las distancias mínimas
Conjunto test y matriz M	0.275
Todo metal y matriz M	0.68
Todo clásico y matriz M	0.55
Todo metal-clásico y matriz M	0.61

Tabla5.2. Medias de las distancias mínimas obtenidas.

Como podemos observar en la Tabla5.2, con esta estrategia ECOC sin adaptación online conseguimos replicar la misma hipótesis que en el apartado anterior. El conjunto de test tiene una distancia pequeña y todo clásico-metal tienen una distancia lejana ya que no hay nada aprendido de ellos.

Ahora, solamente falta ver si somos capaces de aprender una clase nueva sin tocar la estructura que hemos aprendido en este apartado. El próximo, y último experimento, se basa en demostrar si la estrategia ECOC es verdaderamente escalable, como hemos explicado anteriormente, y que nos dé resultados correctos.

5.5 Evaluación ECOC online

Para el último experimento, que lo realizaremos en dos partes, utilizaremos la misma estrategia ECOC que en el apartado anterior, misma codificación, pero, como hemos visto que las canciones de metal-clásica dan una distancia lejana, podemos decidir incluir nuevo aprendizaje, es decir, incluir una clase nueva. Para la decodificación utilizaremos la fórmula comentada en el apartado 4.2 ECOC online (Fig5.3), con distancia de Hamming, para calcular las nuevas distancias mínimas.

$$d(P,\ M_i) = \frac{\sum_{0}^{n} |M_{in}| \cdot FD(M_{in}, P(n))}{\sum_{0}^{n} |M_{in}|}$$

Fig5.3. Fórmula a utilizar para realizar en la ECOC.

```
[[ 1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1.  1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1.  1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1.  1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1.  1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1.  1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1.  1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.  1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1. -1.  1. -1.]
 [-1. -1. -1. -1. -1. -1. -1. -1. -1.  1.]]

[[ 1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1.  1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.  1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1. -1.  1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1. -1. -1.  1. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]]
```

Fig5.4. Arriba matriz M apartado anterior, abajo matriz M con clase y clasificador nuevo.

Para la primera parte, lo que haremos es añadir el 90% de las canciones de metal, a aprender, y utilizaremos el 10% de test para comprobar que las distancias mejoran. Para ello, tenemos que añadir a la matriz M una fila nueva con la palabra clave de la nueva clase y una columna nueva con el nuevo clasificador SVM lineal que aprenda sobre los datos nuevos (90% de metal). En la Fig5.4, en la tabla de debajo, podemos ver como quedaría la matriz. Ahora, realizamos la decodificación y podremos ver qué ocurre con las distancias. En la Fig5.5, podemos ver las diferentes distancias obtenidas de cada canción.

```
[0.     0.     0.0909 0.     0.     0.     0.0909 0.     0.     0.
 0.     0.     0.0909 0.     0.0909 0.0909 0.     0.     0.     0.
 0.0909 0.     0.     0.     0.0909 0.     0.     0.0909 0.     0.
 0.     0.0909 0.0909 0.     0.     0.0909 0.     0.     0.     0.    ]
```

(a)

```
[0.0909 0.     0.0909 0.     0.     0.     0.0909 0.0909 0.0909 0.    ]
```

(b)

```
[0.     0.0909 0.1818 0.0909 0.     0.0909 0.     0.0909 0.0909 0.
 0.0909 0.     0.0909 0.0909 0.     0.0909 0.     0.     0.0909 0.
 0.     0.0909 0.0909 0.0909 0.0909 0.0909 0.     0.0909 0.0909 0.
 0.0909 0.0909 0.0909 0.0909 0.     0.0909 0.     0.     0.     0.0909
 0.     0.     0.     0.0909 0.0909 0.0909 0.0909 0.     0.     0.0909
 0.0909 0.0909 0.0909 0.0909 0.     0.0909 0.     0.     0.0909 0.
 0.     0.     0.0909 0.0909 0.0909 0.0909 0.0909 0.0909 0.0909 0.0909
 0.0909 0.0909 0.     0.     0.     0.     0.0909 0.0909 0.     0.0909
 0.0909 0.0909 0.     0.     0.0909 0.     0.0909 0.     0.     0.
 0.     0.     0.0909 0.     0.     0.     0.     0.     0.0909 0.    ]
```

(c)

Fig5.5. **(a)** Distancias mínimas de cada canción de test a la matriz M. **(b)** Distancias mínimas de 10% metal a la matriz M. **(c)** Distancias mínimas de toda las canciones de clásica a la matriz M.

Distancia realizada entre	Media de las distancias mínimas
Conjunto test y matriz M	0.0250
10% metal y matriz M	0.0455
Todo clásico y matriz M	0.0500

Tabla5.3. Medias de las distancias mínimas obtenidas.

En la Tabla5.3, podemos observar que, aprendiendo una clase nueva sin retocar la información aprendida previamente, el conjunto de test sigue con valores bajos y que los de tipo metal, que es lo nuevo que se ha aprendido, la media de la distancia mínima ha bajado y, además, los de tipo clásica, que todavía no se ha aprendido sobre ellos, siguen con distancias más largas.

Entonces, para la segunda parte, realizamos el mismo experimento, pero añadiendo esta vez 90% de clásica a aprender y 10% lo usaremos para comprobar. Para ello, volveremos a añadir otra fila, para guardar la nueva palabra clave, y otra columna, para guardar el nuevo clasificador binario que aprenda sobre estos datos nuevos. Volvemos a realizar la decodificación y mostraremos los resultados obtenidos. En la Fig5.6 podemos ver como queda la matriz y en la Fig5.7 podemos ver las distancias obtenidas.

```
[[ 1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1. -1.  1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1. -1. -1.  1. -1. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]]
```

Fig5.6. Matriz M añadiendo nuevo aprendizaje.

```
[0.     0.     0.0833 0.     0.     0.0833 0.0833 0.     0.     0.
 0.     0.     0.0833 0.     0.     0.0833 0.     0.     0.     0.
 0.0833 0.     0.     0.     0.0833 0.     0.     0.0833 0.     0.
 0.     0.     0.0833 0.     0.     0.0833 0.0833 0.     0.     0.    ]
```

(a)

```
[0.0833 0.     0.0833 0.     0.     0.     0.0833 0.0833 0.0833 0.    ]
```

(b)

```
[0.0545 0.     0.098  0.     0.     0.     0.0362 0.0053 0.0648 0.    ]
```

(c)

Fig5.7. **(a)** Distancias mínimas de cada canción de test a la matriz M. **(b)** Distancias mínimas del 10% metal del apartado anterior a la matriz M. **(c)** Distancias mínimas del 10% de clásica a la matriz M.

Distancia realizada entre	Media de las distancias mínimas
Conjunto test y matriz M	0.0229
10% metal y matriz M	0.0417
10% clásica y matriz M	0.0259

Tabla5.4. Medias de las distancias mínimas obtenidas.

Viendo los resultados obtenidos de la Tabla5.4, observamos que ahora que hemos vuelto a aprender una nueva clase, el conjunto de test y el 10% de metal siguen con valores muy similares al anterior apartado, en cambio, con las canciones de clásica, vemos que ahora, que ya hay aprendizaje sobre ellos, las distancias son muy menores a la primera parte de este experimento. Con todo esto, hemos verificado que ECOC es un sistema escalable, que nos permite crear clases nuevas sin tener que re-calcular todas las clases ya creadas.

6 Conclusiones

Nuestro objetivo era desarrollar un sistema de aprendizaje no supervisado escalable, es decir, que sea capaz de clasificar correctamente y si aparece un tipo de dato nuevo, ser capaz de actualizarse para aprender sobre ellos y poder clasificarlos.

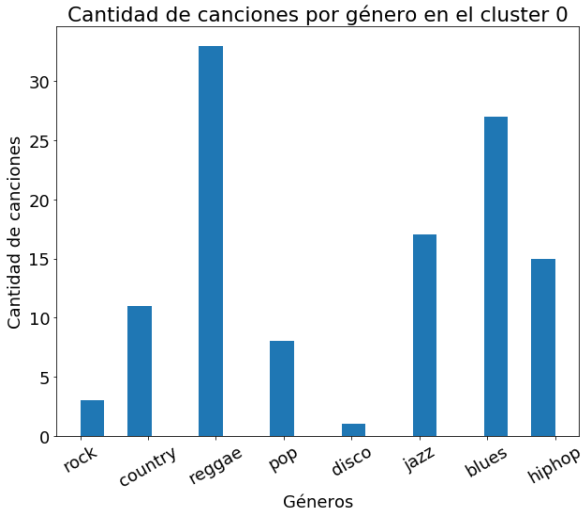
Para ello, utilizamos clustering jerárquico, y, como hemos visto en el apartado anterior, conseguimos cumplir el objetivo marcado, ser capaces de clasificar las canciones y crear nuevos clusters si fuese necesario, pero nos surgió el problema de que para aprender un cluster nuevo teníamos que volver a crear todos los clusters cada vez.

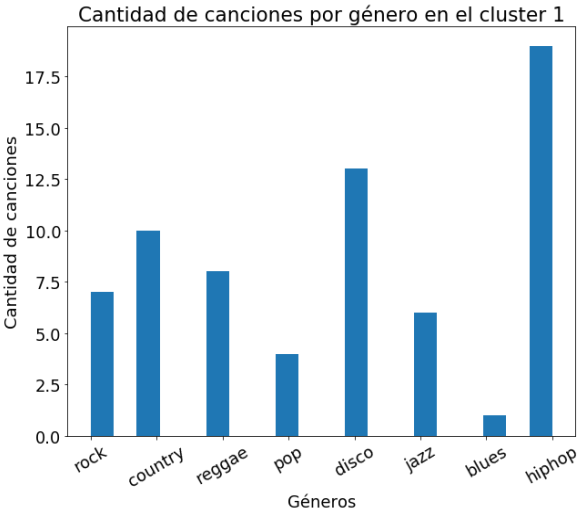
Para solucionar el problema, utilizamos las estrategias ECOC. Primero, vimos que utilizando ECOC básico llegábamos a las mismas hipótesis que clustering jerárquico. Finalmente, esta estrategia nos ha permitido, sin re-calcular lo aprendido previamente, aprender clases nuevas. En los experimentos del apartado anterior, hemos podido comprobar que efectivamente ECOC online funciona, ya que cuando se aprende una clase nueva, conseguimos reducir las distancias mínimas de estas.

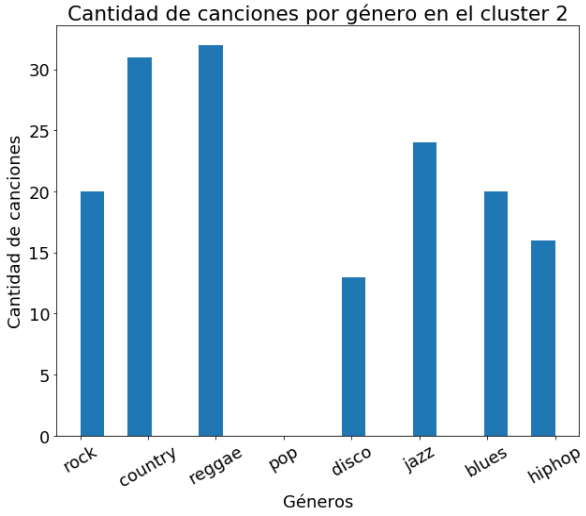
Gracias a la mezcla de clustering jerárquico y el ensamblado ECOC online ya tendríamos nuestro sistema no supervisado escalable. Además, de la misma forma que ECOC nos permite añadir información nueva, esta estrategia también nos permitiría quitar información, de forma online, que ya no se utilice. Ya sea porque nunca se clasifica en una clase o porque nos han comentado que una clase ya no se utiliza, podríamos eliminar esa clase que simplemente ocupa espacio en memoria para nada, pero este objetivo no estaba dentro de nuestro proyecto.

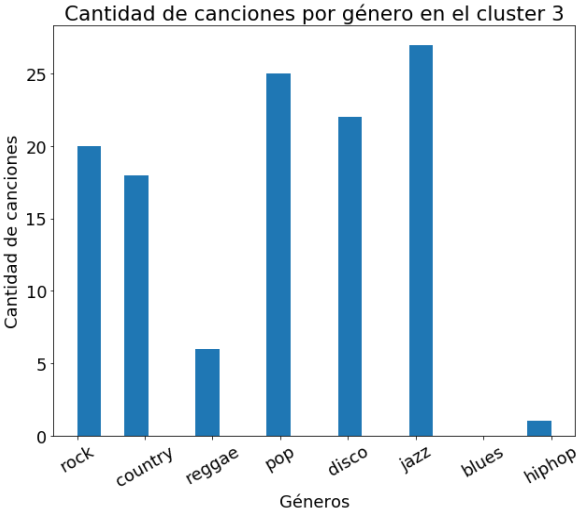
Otro aspecto importante a destacar del ensamblado ECOC es que lo podemos combinar con otras técnicas no comentadas en este trabajo. Por lo tanto, como hemos visto que utilizando extractor de características y clasificadores muy estándar funciona correctamente, de cara al futuro, sería interesante utilizar el mismo ensamblado ECOC, pero con algoritmos más potentes, como por ejemplo, modelos de Deep Learning.

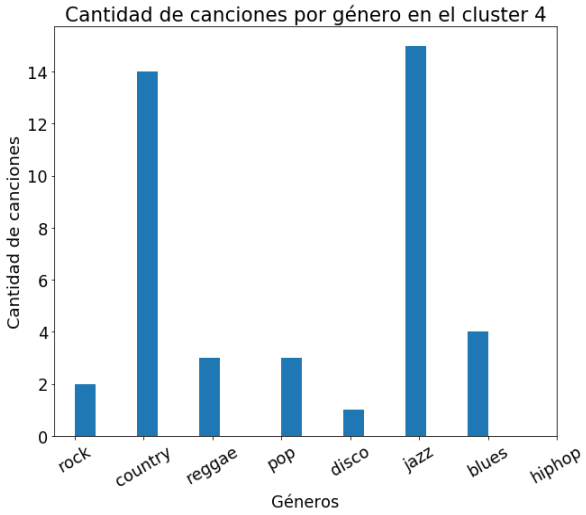
Apéndice 1

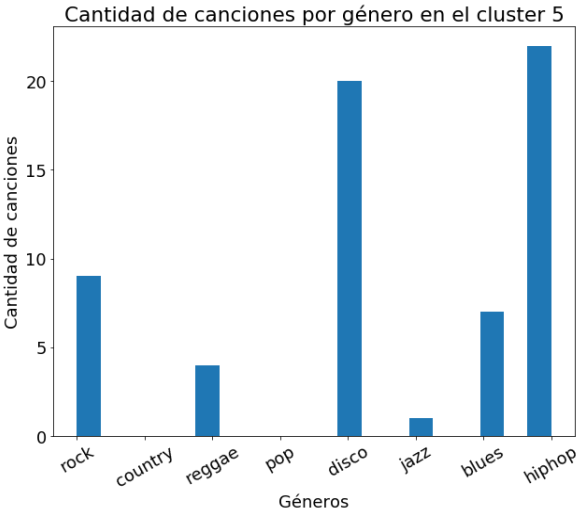

Cantidad de canciones por género en el cluster 0
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop

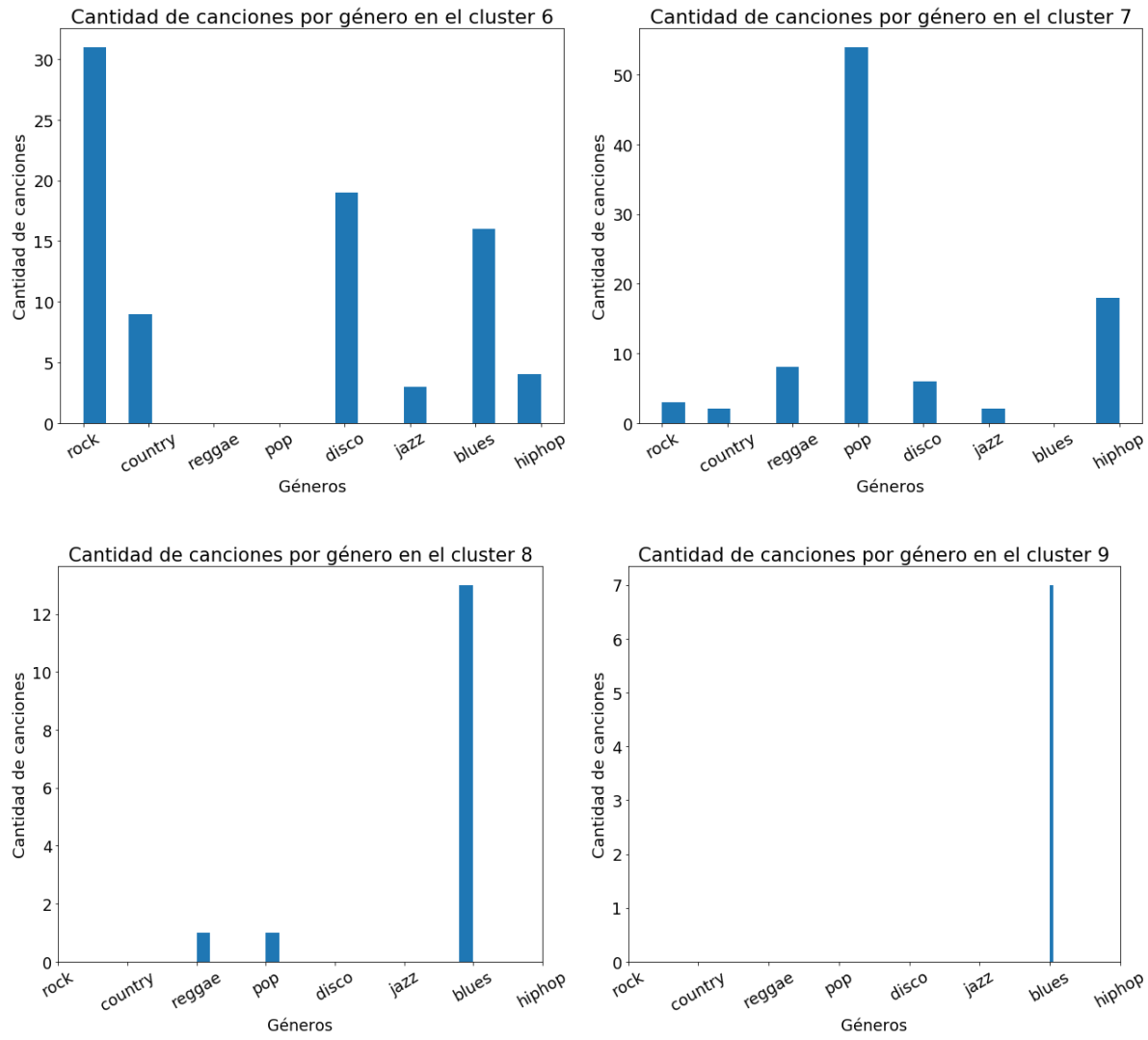

Cantidad de canciones por género en el cluster 1
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop


Cantidad de canciones por género en el cluster 2
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop


Cantidad de canciones por género en el cluster 3
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop


Cantidad de canciones por género en el cluster 4
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop


Cantidad de canciones por género en el cluster 5
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop

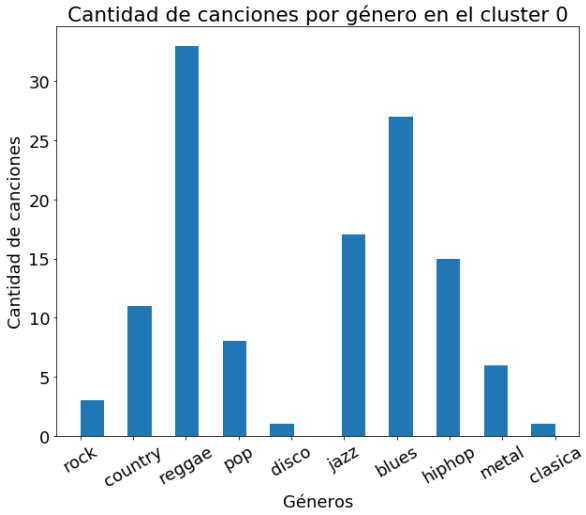


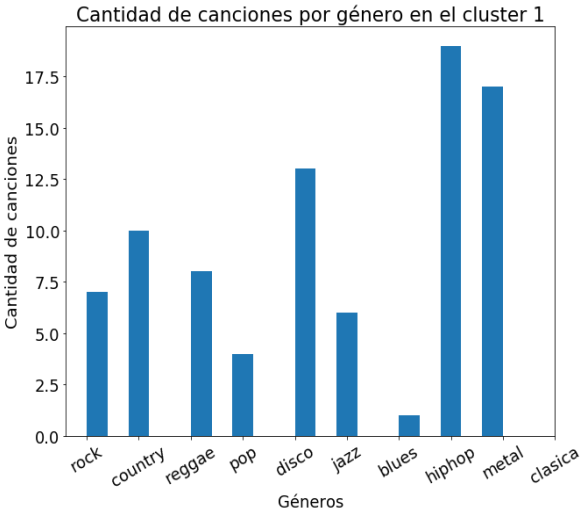
FigA.1. Gráficas de los 10 clusters generados.

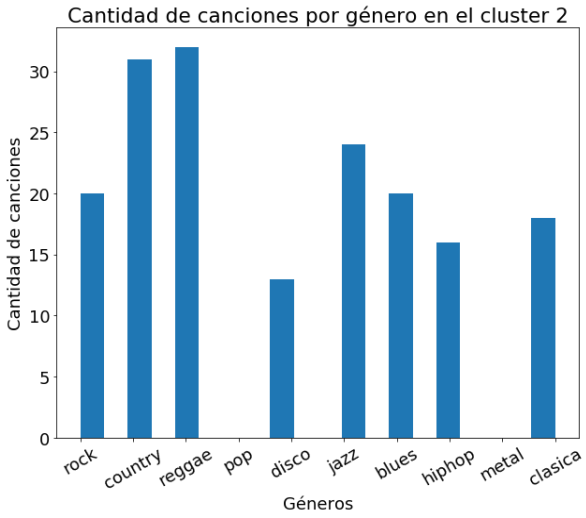
Con el algoritmo clustering jerárquico creamos 10 clusters. En la FigA.1 podemos ver la cantidad de canciones de cada género que se agrupan en cada cluster. Como podemos observar, los clusters no están distribuidos según el género de cada audio, sino, más bien, por la semejanza de características musicales como ritmos similares, mismos instrumentos, notas musicales similares, etc.

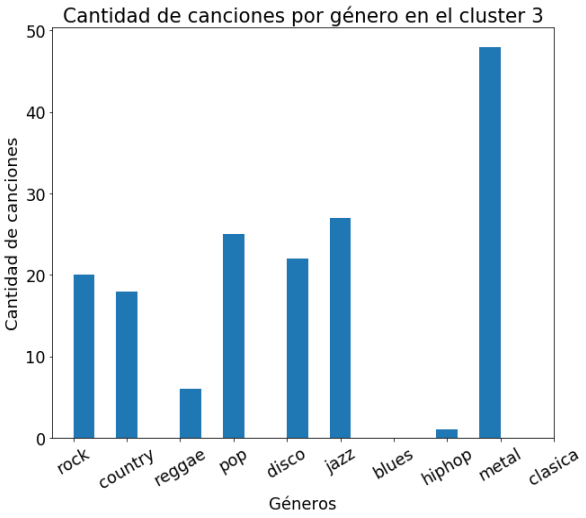
Si observamos algunos clusters, como por ejemplo el cluster 5, donde destaca la música Disco y el Hip Hop, estos dos géneros suelen tener el mismo tempo parecido, es decir, tienen similitudes en la velocidad con la que se reproducen. En el cluster 6, vemos que destaca el Rock junto con Disco, Blues y un poco de Country, estos géneros pueden llegar a parecerse ya que Rock proviene del Blues, por lo tanto tienen características similares, y tanto Disco como Country, en algunos casos, tiene tempos parecidos al Rock. En el cluster 7, observamos que destaca solamente el Pop, con muy pocas canciones del resto, porque siempre se realizan con el mismo tempo, y algunas de otros géneros tienen ese tempo y por eso las agrupa en el mismo cluster.

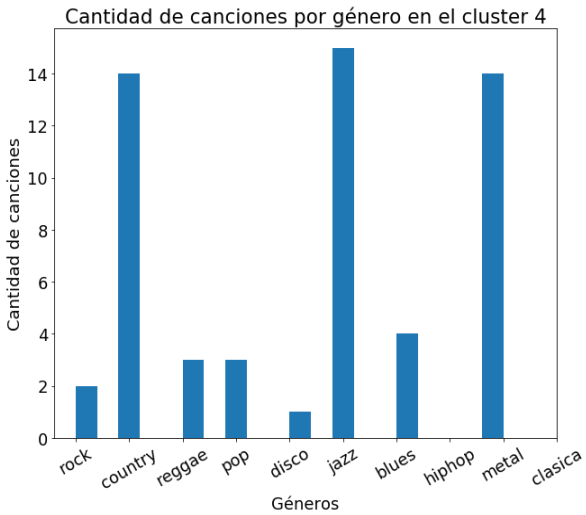
Apéndice 2

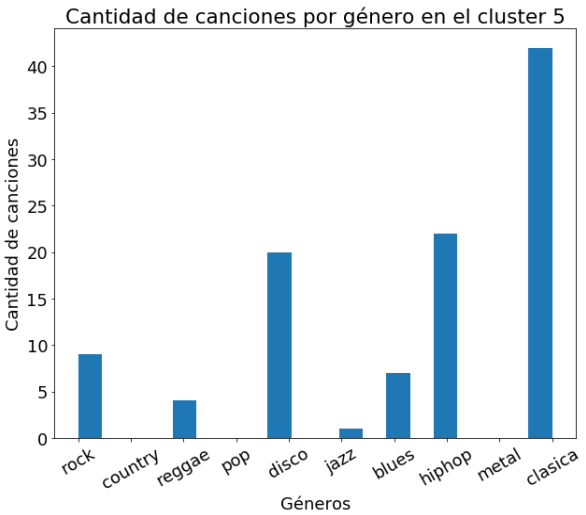

Cantidad de canciones por género en el cluster 0
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop
metal
clasica

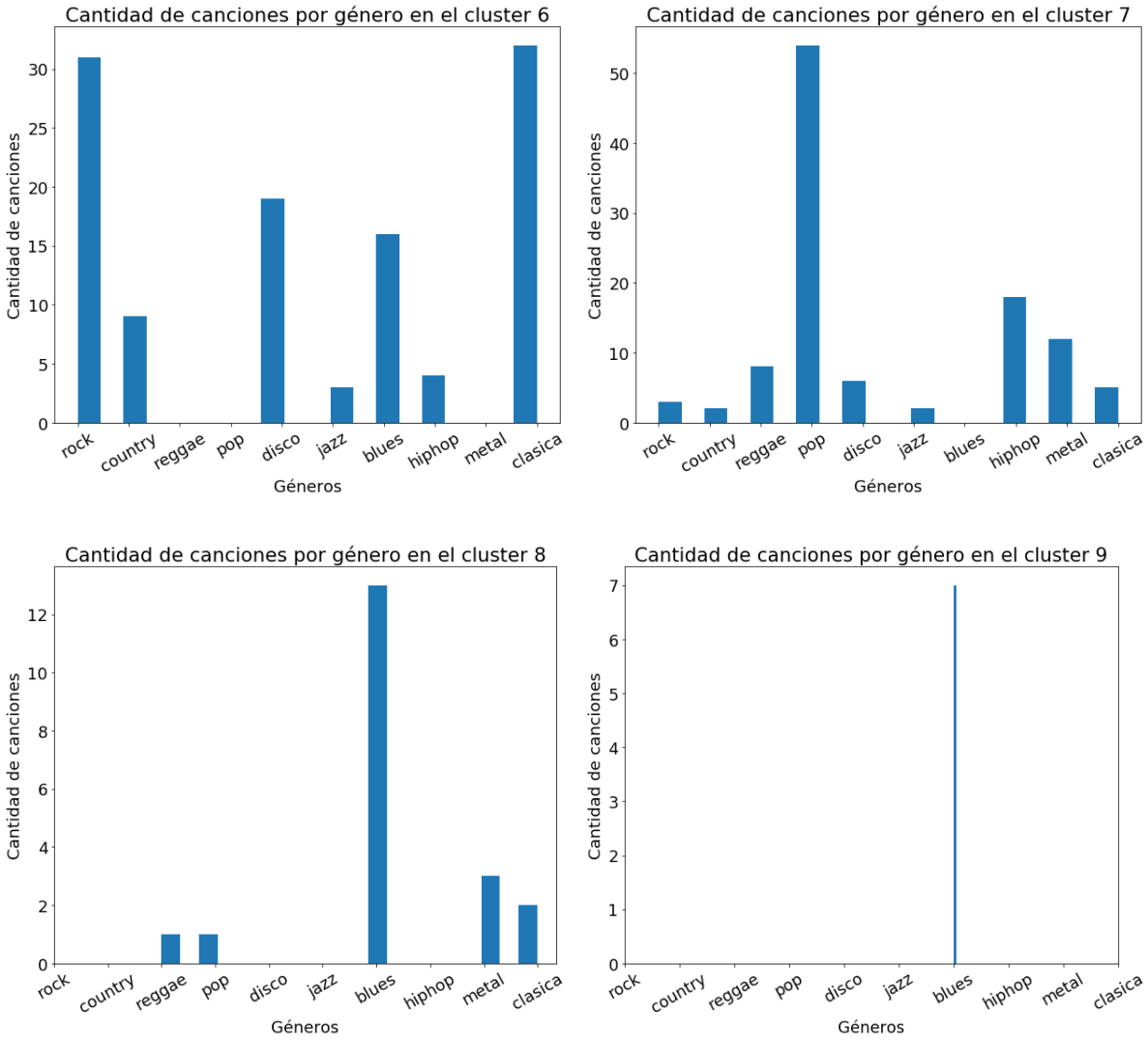

Cantidad de canciones por género en el cluster 1
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop
metal
clasica


Cantidad de canciones por género en el cluster 2
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop
metal
clasica


Cantidad de canciones por género en el cluster 3
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop
metal
clasica


Cantidad de canciones por género en el cluster 4
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop
metal
clasica


Cantidad de canciones por género en el cluster 5
Cantidad de canciones
Géneros
rock
country
reggae
pop
disco
jazz
blues
hiphop
metal
clasica



FigA.2. Clasificación de metal y clásica en los 10 clusters con algoritmo KNN aproximado.

Con KNN aproximado, calculamos la distancia mínima entre una canción y los centroides de cada cluster, en vez de por cada elemento de los clusters, y en la FigA.2 podemos observar en qué cluster se clasificarían todas las canciones de metal y clásica aunque tengan una distancia lejana.

Referencias

[1] Music Information Retrieval. State of the art: Audio tag classification (Genre). 2017
https://musicinformationretrieval.wordpress.com/2017/02/06/state-of-the-art-audio-tag-classification-genre/

[2] Miguel Flores. Deep Music Genre. Stanford University. 6 págs.
http://cs231n.stanford.edu/reports/2017/pdfs/22.pdf

[3] Deep Sound
http://deepsound.io/music_genre_recognition.html

[4] Marsyas
http://marsyas.info/downloads/datasets.html

[5] S. Gupta, J. Jaafar, W. Fatimah y A. Bansal. Feature Extraccion Using MFCC. University Tecknologi PETRONAS, Malasia. 8 págs. 2013.
https://aircconline.com/sipij/V4N4/4413sipij08.pdf

[6] MIR Lab
http://mirlab.org/jang/books/audioSignalProcessing/speechFeatureMfcc.asp?title=12-2%20MFCC

[7] PennState Eberly College of Science
https://online.stat.psu.edu/stat505/lesson/11

[8] Sebastian Raschka. Implementing a PCA in python, step by step. 2014.
https://sebastianraschka.com/Articles/2014_pca_step_by_step.html

[9] Ciencia & Datos. Aprendizaje no supervisado en machine learning: Agrupación. 2019.
https://medium.com/datos-y-ciencia/aprendizaje-no-supervisado-en-machine-learning-agrupaci%C3%B3n-bb8f25813edc

[10] Lidgi Gonzalez. Todo sobre aprendizaje no supervisado. 2018.
https://ligdigonzalez.com/todo-sobre-aprendizaje-no-supervisado-en-machine-learning/

[11] Jacob Avila Camacho. Algoritmos, Inteligencia Artificial, Machine Learning. 2020.
https://www.jacobsoft.com.mx/es_mx/clustering-jerarquico-con-python/

[12] Joaquín Amat Rodrigo. Clustering y Heatmaps: aprendizaje no supervisado. 2017.
https://rpubs.com/Joaquin_AR/310338

[13] Universidad de Granada.
https://www.ugr.es/~gallardo/pdf/cluster-3.pdf

[14] S. Escalera, David M, E. Puertas, P. Radeva, O. Pujol. Online error correcting output codes. 10 págs. 2011.

[15] S. Escalera, David M.J. Tax, O. Pujol, P. Radeva, Member, IEEE y Robert P. W. Subclass Problem-Dependent Design for Error-Correcting Output Codes. 14 págs. 2008.

[16] Librería LibROSA
https://librosa.org/librosa/

[17] Librería Scikit-Learn
https://scikit-learn.org/stable/modules/classes.html