Recomendación de Música Basada en Contenido con Deep Learning

Facultad de Matemática, Astronomía, Física y Computación

Universidad Nacional de Córdoba

Francisco Carranza

Director: Dr. Franco M. Luque

Abstract

Resumen*: La música es parte de nuestras vidas. Cada vez consumimos más contenido digital, queremos todo al instante y acorde a nuestros gustos. Dado el gigantesco catálogo musical que existe, resulta indispensable contar con un sistema automático que recomiende canciones relevantes para los usuarios. En este trabajo implementaremos un sistema de recomendación de música siguiendo estrategias del estado del arte en aprendizaje automático. Dado un historial de reproducciones de cada usuario, construiremos una matriz de baja dimensionalidad que modelará vectores de características latentes de las canciones. Luego, para poder recomendar nuevas canciones no incluidas en la base de datos original, intentaremos predecir esos vectores latentes a partir del audio crudo. Para ello transformaremos el audio en espectrogramas y usaremos esta representación como entrada de un modelo de aprendizaje automático profundo. Finalmente recomendaremos nuevas canciones teniendo en cuenta la similaridad entre vectores.*

Palabras claves: Sistema de recomendación, aprendizaje automático, aprendizaje profundo, redes neuronales convolutivas, audio, música, espectrogramas, mfcc.

***Abstract**: Music is part of our lives. Nowadays we consume more digital content, we want everything instantly and according to our tastes. Given the gigantic existing musical catalog, it is essential having an automatic system that recommends relevant songs to users. In this work we will implement a music recommendation system following state of the art techniques in machine learning. Given a real dataset of user play-count, we will build a matrix of low dimensionality that model songs as a latent characteristics vectors. Then, to be able to recommend new songs not included in the original dataset, we predict those latent vectors from raw audio. In this task, we will transform the audio into spectrograms and use this representation as input of a deep machine learning model. Finally we recommend new songs taking into account the similarity between vectors.*

Keywords: Recommendation system, machine learning, deep learning, convolutional neural networks, audio, music, spectrograms, mfcc.

Agradecimientos

Agradezco a mi familia, por haberme acompañado y motivado todos estos años.

A mi director Franco Luque y a mi consejero Diego Tomassi. Por la cantidad de reuniones, videollamadas y dudas resueltas.

A los profesores y a la comunidad FaMAF. Por haberme brindado una excelente formación académica.

Índice

1. Introducción

En la actualidad las personas consumen mucho contenido multimedia: Imágenes, videos, películas, series, música. La tecnología ha democratizado el acceso y ha hecho que sea instantáneo. Se abrió mucho el mercado y cualquier persona con conexión a internet puede consumir estos recursos. Dada la enorme oferta de productos, se invierte mucho en sistemas personalizados para mejorar la experiencia de usuario. Nos movemos mucho y **queremos que las cosas se nos entreguen rápido.** Tampoco nos gusta pasar demasiado tiempo en un mismo sitio web o servicio. Por eso, es de vital importancia, anticiparse y mostrar los productos que al cliente le gustarán.

Tomemos algunos ejemplos conocidos: Amazon, Mercado Libre, Netflix, Spotify. En todos ellos existe un perfil personalizado que a medida que vamos comprando y consumiendo, deja huella de lo que nos gusta y esa información se aprovecha para sugerirnos nuevo contenido relevante.

En particular, Spotify es un servicio de streaming de música y que lidera el mercado con más de 140 millones de usuarios y con un catálogo de 30 millones de canciones. La idea de este trabajo surgió de utilizar una de sus funciones: La creación de listas de reproducción. A medida que se van agregando canciones a ésta lista, el sistema recomienda canciones de acuerdo a la similaridad de las agregadas y las preferencias del usuario..

El sistema de recomendación más utilizado por excelencia, y que da mejores resultados, es el basado en información de uso de los usuarios. A través de grupos similares de personas podemos establecer patrones de consumo y recomendar ítems que mejor respondan a estas características. La gran desventaja de este enfoque es que cuando no tenemos historial de uso, ya sea porque el ítem es muy nuevo o porque no es popular, el recomendador no funciona bien. A esto se le llama problema de *cold start*. En este caso,

proponemos un sistema de recomendación basado en **contenido**; es decir, en establecer patrones sobre las características de audio.

1.1 Referencia principal

Tomamos como referencia principal al trabajo: *Deep content-based music recommendation (van den Oord, Dieleman, and Schrauwen)*. En él, desarrollan un sistema que utiliza el enfoque colaborativo como *ground-truth* y luego junto con el contenido de cada canción, construye un sistema de recomendación basado en audio que soluciona el problema de *cold start*.

Nuestro trabajo desarrolla las ideas de Sander. Primero veremos cómo extraer información útil a partir de un historial de reproducciones de usuarios. Modelaremos a las canciones como vectores de factores latentes representando características ocultas del comportamiento de los usuarios. Para contrarrestar el problema de *cold start*, implementaremos un sistema que a partir del audio crudo de una canción, predice su vector correspondiente de factores latentes. Al final, tendremos un sistema que puede recomendar canciones similares teniendo en cuenta el audio.

1.2 Descripción breve de los capítulos

En el **Capítulo 2** daremos una breve base teórica de los conceptos que veremos más adelante. Hablaremos del dominio de la extracción de features sobre música, de las propiedades del sonido y su representación digital y por último, de las técnicas de aprendizaje automático que utilizaremos.

En el **Capítulo 3** introduciremos a los sistemas de recomendación y más específicamente a la recomendación de música. Contaremos cuáles son los distintos enfoques y cómo uno de ellos ayuda a solucionar un problema clave del

otro. Luego listaremos los conjuntos de datos que son empleados para correr los experimentos.

En el **Capítulo 4**, detallaremos un algoritmo de filtrado colaborativo para matrices de ítem-cantidad de reproducciones. Explicaremos cómo las canciones representan vectores a partir de los datos de uso generado por los usuarios.

El **Capítulo 5** mostrará los modelos de aprendizaje automático utilizados para predecir los vectores generados (a partir de los datos de uso) utilizando representaciones digitales sonoras. Junto al anterior, son los capítulos más importantes del trabajo.

En el **Capítulo 6** resolveremos un problema dentro del dominio de la música, la clasificación de género.

En el **Capítulo 7** daremos una conclusión de los resultados del trabajo. Hablaremos del recomendador y de sus enfoques implementados.

Finalmente en el **Capítulo 8** dejamos la puerta abierta para seguir mejorando el sistema, enumerando algunos caminos a seguir.

2. Preliminares

En este capítulo introduciremos conceptos básicos pero importantes a la hora de entender el problema y la solución planteada.

Primero, describiremos el dominio de la música, los métodos utilizados y por último, algunas aplicaciones y desafíos a la que se enfrenta.

Luego, veremos qué es el audio y cómo se representa digitalmente, las propiedades del sonido y dos representaciones del mismo: Espectrogramas y MFCCs.

En la tercera parte, mencionaremos brevemente las técnicas de aprendizaje automático más utilizadas y aquellos algoritmos que aplicaremos en este trabajo.

Finalmente, explicaremos en detalle, los sistemas de recomendación en general y la recomendación sobre el dominio de la música, sus dos enfoques, el problema de uno y cómo el otro ayuda a resolverlo. Al final, hablaremos de los conjuntos de datos sobre los que construimos los modelos.

2.1 Music Information Retrieval

Music Information Retrieval (MIR) es una ciencia interdisciplinaria que extrae **información** a partir de **música**. MIR es un campo pequeño pero en crecimiento y con muchas aplicaciones reales. Reúne a investigadores de diferentes disciplinas tales como: Musicología, psicoacústica, psicología, estudios académicos de música, procesamiento de señales, informática, aprendizaje automático o alguna combinación de estas.

Los métodos utilizados por MIR pueden clasificarse en los siguientes ejes:

1. **Fuente de datos:** Las ***partituras*** dan una descripción clara y lógica de la música para trabajar, pero el acceso a partituras, ya sean digitales o no, es

muy limitado. La música MIDI también ha sido utilizada por razones similares, pero la conversión a MIDI desde cualquier formato conlleva la pérdida de datos, a menos que la música haya sido escrita teniendo en cuenta los estándares MIDI, lo cual es raro.

Los formatos de **audio digital** como WAV, mp3, y ogg son utilizados cuando el audio es parte del análisis. Formatos comprimidos con pérdida como mp3 y ogg son adecuados para el oído humano, pero pueden obviar datos cruciales para el estudio. Además algunas codificaciones crean artefactos que podrían inducir errores a cualquier analizador automático. A pesar de esto, la ubicuidad del formato mp3 ha implicado que muchas de las investigaciones desarrolladas en el campo lo utilicen como material de fuente.

Cada vez se emplean más en MIR los **metadatos** obtenido de la web para un mejor entendimiento de la música dentro de su contexto cultural. Esto recientemente incluye el análisis de etiquetas sociales para música.

2. **Representación de atributos:** El análisis a menudo puede requerir resumir información, y para la música (como con muchas otras formas de datos) esto se logra mediante la extracción de atributos, especialmente cuando se está analizando el contenido del audio y se desea aplicar aprendizaje automático. El propósito es reducir la gran cantidad de datos a un conjunto de valores gestionable de modo que el aprendizaje pueda llevarse a cabo en un marco de tiempo razonable. Una característica que se extrae comúnmente son los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC). Otros atributos pueden ser empleados para representar la nota musical, los acordes, las armonías, la melodía, el tono principal, los compases por minuto o el ritmo en la pieza.

3. **Estadística y aprendizaje automático**: Métodos computacionales para clasificación, clustering, y extracción de atributos musicales para música mono- y polifónica. Métodos para determinar la semejanza entre piezas musicales. Bases de datos y métodos formales para la identificación y reconocimiento automatizados de música, filtrado de música y consultas

de música, lenguajes de consulta y metadatos o protocolos para el manejo de información relacionada con la música. Métodos de recuperación con búsqueda distribuida y sistemas multiagentes. Software para recuperación de música empleando la Web Semántica, agentes inteligentes, software colaborativo, búsquedas web, búsqueda por tarareo y huella acústicas. Análisis de música y representación de conocimiento, resumen automático, modelos formales de música y partituras digitales e indexación de música y sus metadatos.

Estas son algunas aplicaciones que podemos encontrar:

- **Transcripción automática de música:** Es el proceso de convertir un registro de audio a notación simbólica, como una partitura o un archivo MIDI. Este proceso implica varias tareas que incluyen: detección de tonos, detección del inicio de una nota musical o sonido, estimación de la duración, identificación de instrumentos y extracción de información rítmica. Esta tarea se vuelve más difícil con un número mayor de instrumentos y un mayor nivel de polifonía.

- **Separación de pistas y reconocimiento de instrumentos:** La separación de pistas consiste en extraer las pistas originales tal como fueron grabadas. Cada una de estas pistas puede tener registrado más de un instrumento. El reconocimiento de instrumentos consiste en identificar aquellos que se emplearon en una grabación y/o en separar cada instrumento en una pista distinta. Así es como se crean pistas de karaoke a partir de pistas normales. Este proceso aún no está perfeccionado dado que la voz ocupa parte del espacio de frecuencia de otros instrumentos.

- **Categorización automática:** La tarea de clasificación consiste en categorizar ejemplos en un conjunto de clases asignando una o más etiquetas a cada uno de ellos. Ésta tarea puede llevarse a cabo manualmente, como por ejemplo el proyecto Pandora Genome (John) que

mencionaremos más adelante, pero en general requiere de mucho tiempo y recursos. Los problemas de clasificación más populares dentro de MIR son:

- **Clasificación de instrumentos:** Es el problema menos desafiante ya que el sonido que producen afecta al espectro de frecuencias y es lo que nosotros percibimos como *timbre*.
- **Clasificación de género:** El concepto de género, de alguna manera no está bien definido y típicamente se manifiesta en la señal de audio de diferentes maneras (instrumentos, patrones rítmicos, melodía, etc) y en diferentes escalas de tiempo. Es un problema más complicado que el anterior pero que sigue siendo muy popular dentro de MIR. Un dataset muy conocido es GTZAN (Sturm), un total de 1000 audios de canciones divididas en 10 géneros. Como dijimos antes, es difícil encontrar datasets lo suficientemente grandes y de calidad para hacer experimentos, por lo que GTZAN es ampliamente utilizado para ésta tarea.
- **Clasificación de estado anímico o mood*:*** El estado anímico de una canción es todavía más ambiguo y difícil de definir que el género. Sin embargo, hay mucho interés en esta tarea debido a la recomendación teniendo en cuenta el contexto del oyente.
- **Otras tareas:** Podemos encontrar: Clasificación de artistas, identificación de tono (de toda una pieza musical o de un acorde) o el tempo.

- **Generación de música:** Utilizando técnicas de Deep Learning se han desarrollado sistemas capaces de generar música a partir del aprendizaje de canciones. Ejemplos son: *Deep Jazz (Kim), BachBot (Liang), WaveNet (Van Den Oord et al.)* pero que no logran convencer mucho todavía. Recientemente se investigó cómo el tempo y la expresividad refleja la belleza de una pieza pianística. Utilizando *Performance RNN (Simon and*

Oore) se logró una notable diferencia a la hora de escuchar música clásica generada automáticamente.

- **Recomendación de música:** Su objetivo es recomendar nuevos artistas y nuevas canciones a los oyentes tomando como base sus preferencias. Estas preferencias pueden determinarse de muchas maneras y de diferentes fuentes de datos. Consecuentemente, existen muchas maneras de construir sistemas de recomendación de música. En general, los recomendadores se han vuelto muy populares en los últimos años y los podemos encontrar en otros dominios como: Libros, películas, productos, turismo e incluso personas (citas online).

2.2 Audio

El sonido, en física, es cualquier fenómeno que involucre la propagación de ondas mecánicas (sean audibles o no), generalmente a través de un fluido (u otro medio elástico) que esté generando el movimiento vibratorio de un cuerpo. El sonido humanamente audible consiste en ondas sonoras y ondas acústicas que se producen cuando las oscilaciones de la presión del aire, son convertidas en ondas mecánicas en el oído humano y percibidas por el cerebro. La propagación del sonido es similar en los fluidos, donde el sonido toma la forma de fluctuaciones de presión.

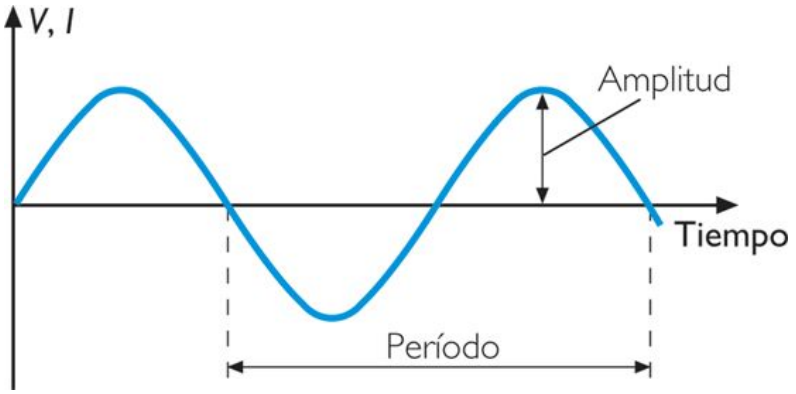

Figura: Onda de sonido.

Propiedades

Las cuatro cualidades básicas del sonido son la altura, la duración, la intensidad y el timbre o color.

Cualidad	**Característica**	**Rango**
Altura o tono	Frecuencia de onda	Agudo, medio, grave
Duración	Tiempo de vibración	Largo o corto
Intensidad	Amplitud de onda	Fuerte, débil o suave
Timbre	Armónicos de onda o forma de la onda	Depende de las características de la fuente sonora

Las señales de audio representan ondas de presión, son patrones de movimiento de altas y bajas presiones. Son continuas tanto en altitud como en tiempo. Ahora, para almacenarlas en formato digital hay que cuantizar ambas dimensiones. Este procedimiento se llama *pulse-code modulation* o PCM. El resultado es un vector de enteros que representa una aproximación de la onda real. El proceso de cuantización o *sampling* implica una discretización de las frecuencias y es donde se introduce pérdida de información.

El formato ideal es uno **sin pérdida**, es decir, que se almacene el sonido tal cual como fue sampleado. Un ejemplo es el formato WAVE. Pero el más popular es el MP3, un formato **con pérdida**, que remueve información sonora que el oído humano no puede percibir, logrando un archivo de considerable menor tamaño.

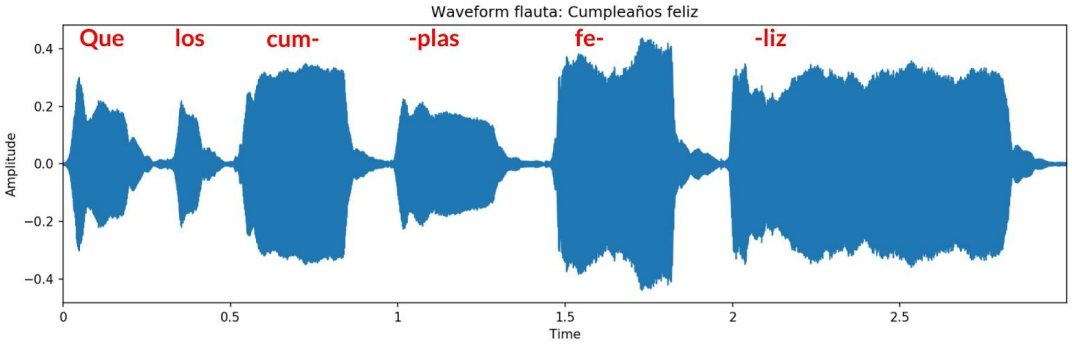

Figura: Fragmento de la canción feliz cumpleaños tocado en flauta dulce.

En el contexto de MIR, se trabaja sobre *representaciones de mediano-nivel* que se extraen a partir de las señales de audio. Mayormente representan el sonido de la forma en que los seres humanos escuchamos: Patrones de frecuencias que varían según el tiempo. Se dicen *medianas* porque se encuentran entre las de *bajo-nivel* (señal cruda de audio) y *alto-nivel* (género musical por ejemplo).

Las representaciones de mediano nivel más utilizadas son las de tiempo-frecuencia. En vez de describir cómo la amplitud de onda varía en el tiempo, describe mediante bandas, las frecuencias y volúmenes presentes en un momento dado.

Se utilizan mucho los **espectrogramas** como representaciones de mediano-nivel. Un espectrograma de una señal se obtiene computando la *Short-Time Fourier Transformation STFT* de pequeñas ventanas de tiempo que suelen superponerse para lograr mayor suavidad. Sobre estas ventanas transformadas, se obtienen espectros de potencia que indican la energía presente en cada banda de frecuencias en cada ventana. Cada una de estas constituye un cuadro o *frame*. Una sucesión de frames son concatenados en una matriz para formar un espectrograma.

A menudo, los espectrogramas son post-procesados. El algoritmo STFT devuelve una representación lineal del sonido, pero el oído humano percibe el volumen en escala logarítmica. Entonces una opción son los espectrogramas-logarítmicos.

Otra forma muy utilizada en el campo del procesamiento del habla, son los espectrogramas de mel.

Ejemplos de representaciones de alto-nivel son los llamados *Mel Frequency Cepstral Coefficients* MFCCs (Logan and Others) o las representaciones chroma (que describen solo información del tono descartando variaciones tímbricas). El problema de los MFCCs es que pierden información temporal cuando se obtienen de señales de larga duración, lo que no es conveniente cuando tratamos con canciones.

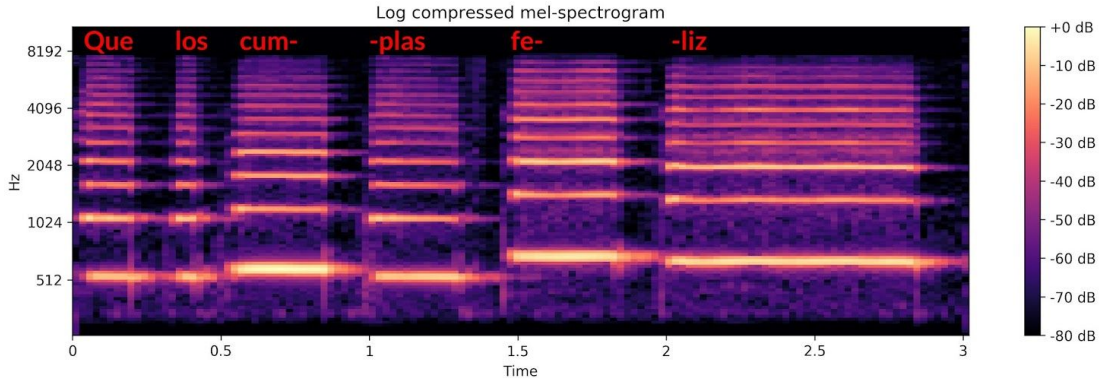

Figura: Espectrograma de mel de fragmento del feliz cumpleaños en flauta.

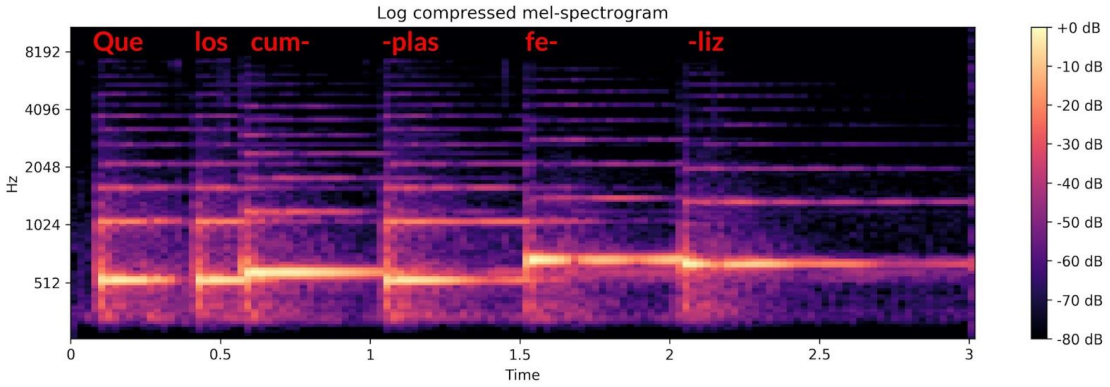

Figura: Esta vez el mismo fragmento de Cumpleaños feliz pero tocado con un piano. Son las mismas notas, mismo tempo, pero el piano produce distintos armónicos que la flauta.

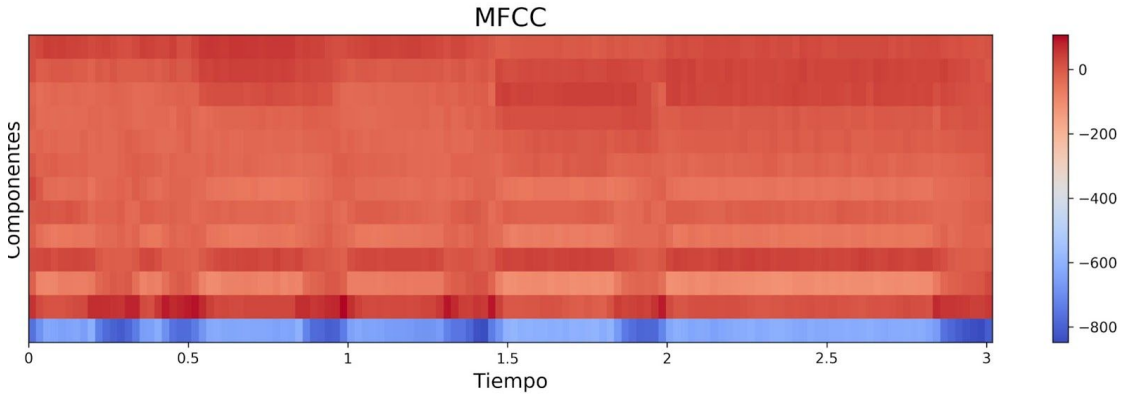

Figura: MFCC 13 componentes del cumpleaños feliz en flauta

2.3 Aprendizaje automático

El aprendizaje automático o aprendizaje de máquinas (del inglés, *Machine Learning*) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento.

El aprendizaje automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, videojuegos y robótica.

2.3.1 Aprendizaje supervisado

En aprendizaje automático y minería de datos, el aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos (normalmente vectores) de datos de entrada y resultados deseados. La salida de la función puede ser un

valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación). El objetivo del aprendizaje supervisado es el de crear una función capaz de **predecir el valor correspondiente** a cualquier objeto de entrada válida después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tiene que ser capaz de generalizar a partir de los datos presentados a las situaciones no vistas previamente.

Ejemplos: Naive Bayes, regresión lineal, SVM, entre otros.

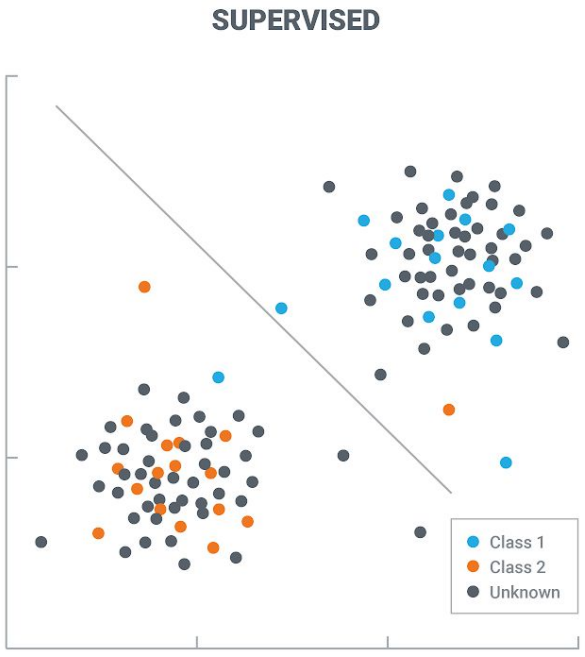

Figura: Ejemplo donde los datos están etiquetados.

2.3.2 Aprendizaje no supervisado

Aprendizaje no supervisado es un método de aprendizaje automático donde un modelo es ajustado a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que **no hay un conocimiento a priori**. En el aprendizaje no supervisado, sólo se trabaja con un conjunto de datos de objetos de entrada. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para

el conjunto de datos. Usa datos históricos que no están etiquetados. El fin es explorarlos para encontrar alguna estructura o forma de organizarlos.

Ejemplos: K-means, Hierarchical clustering, autoencoder, Deep belief network, PCA.

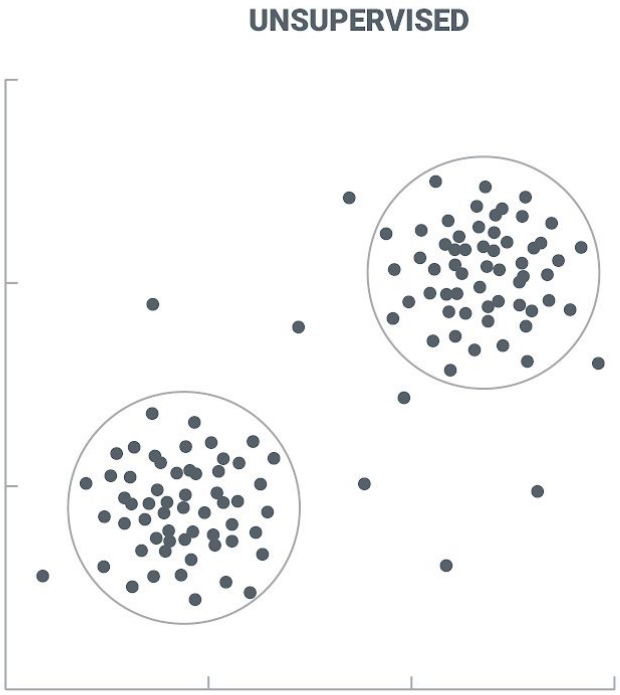

Figura: En este caso, los datos no están etiquetados. El algoritmo se encarga de agruparlos por clases.

2.3.3 Deep learning

Deep Learning es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en los datos, usando arquitecturas compuestas de transformaciones no lineales múltiples.

El aprendizaje profundo es parte de un conjunto más amplio de métodos de aprendizaje automático basados en asimilar representaciones de datos. Una observación (por ejemplo, una imagen) puede ser representada en muchas formas (por ejemplo, un vector de píxeles), pero algunas representaciones hacen más fácil aprender tareas de interés (por ejemplo, "¿es esta imagen una cara

humana?") sobre la base de ejemplos, y la investigación en este área intenta definir qué representaciones son mejores y cómo crear modelos para reconocer estas representaciones.

Varias arquitecturas de aprendizaje profundo, como redes neuronales profundas, redes neuronales profundas convolucionales, y redes de creencia profundas, han sido aplicadas a campos como visión por computador, reconocimiento automático del habla, y reconocimiento de señales de audio y música, y han mostrado producir resultados de vanguardia en varias tareas.

Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

El error cuadrático medio MSE es una función que cuantifica la pérdida de información en la predicción y se corresponde con el valor esperado del error al cuadrado o error cuadrático medio. La diferencia se produce debido a la aleatoriedad o porque el estimador no tiene en cuenta la información que podría producir una estimación más precisa.

2.4 Recomendación de música

Los sistemas de recomendación forman parte de un sistema de filtrado de información, los cuales presentan distintos tipos de temas o ítems de información (películas, música, libros, noticias, imágenes, páginas web, etc.) que son del interés de un usuario en particular. Generalmente, un sistema recomendador compara el perfil del usuario con algunas características de referencia de los temas, y busca predecir el *ranking* o ponderación que el usuario le daría a un ítem que él aún no ha considerado. Estas características pueden basarse en la relación o cercanía del usuario con el tema o con su entorno social.

Estos sistemas típicamente producen una lista de recomendaciones basado en dos enfoques: Filtrado colaborativo y filtrado basado en el contenido. El primero construye un modelo a partir del comportamiento que tuvo el usuario (ítems comprados anteriormente o calificaciones dadas) como también las decisiones que tomaron otros usuarios. El segundo, utiliza características del ítem para recomendar otros ítems con características similares.

Las diferencias de estos enfoques puede verse mediante dos recomendadores populares de música:

- **Last.fm:** Crea estaciones de radio con canciones recomendadas. Observa qué artistas escucha regularmente el usuario y compara esto con el comportamiento de otros usuarios con intereses musicales similares. Es un claro ejemplo de enfoque de filtro colaborativo.

- **Pandora:** Utiliza las características de las canciones o artistas (una colección de 400 atributos construidos manualmente por científicos del *Music Genome Project)*. El *feedback* del usuario (los *likes* por ejemplo) son utilizados para enfatizar atributos cuando un usuario indica "me gusta" o para desenfatizar cuando indica "no me gusta". Este es un claro ejemplo del enfoque basado en contenido.

Explicaremos a continuación cómo nuestro trabajo depende de ambos enfoques.

2.4.1 Enfoque de filtro colaborativo

Es el más utilizado en la industria. Aborda la recomendación teniendo en cuenta sólo el historial de uso de los ítems. La gran ventaja de esto es que es **independiente del contenido**, que puede tratarse de películas, música o productos de una tienda *e-commerce*.

Dentro de este enfoque encontramos dos métodos:

1. **Neighborhood-based:** Se recomiendan canciones escuchadas por otros usuarios con preferencias similares, o canciones similares a las que el usuario ya escuchó.

2. **Model-based:** Tratan de modelar características latentes de usuarios y canciones, las cuales están representadas como **vectores de factores latentes**. Éste método se utilizó en la competencia *The Netflix Prize (Bennett, Lanning, and Others).*

2.4.2 Enfoque basado en el contenido

Podemos recomendar canciones a partir de los **metadatos**, que es información adicional de la canción, tales como artista, album, año de publicación, género, entre otras. Éste método si bien resulta muy fácil de implementar, tiene resultados muy predecibles. Si queremos vender nuevas canciones a un usuario, recomendarle otras de su banda favorita no es muy útil.

Otra forma, es recomendar canciones teniendo en cuenta sus características sonoras. Por ejemplo: Recomendar una canción porque es "parecida" a otras que escucho puede ser muy útil. Algunas características sonoras que se pueden inferir son: Tempo (velocidad del ritmo), ritmo, estado anímico *(mood),* armonía, melodía y muchas otras más. El gran problema es que hay una brecha semántica muy importante entre la señal de audio y el sentimiento que despierta en mí escucharla por ejemplo. Aparte no está definido cómo comparar dos canciones, es decir, qué características mirar a la hora de elegir canciones similares. De más está decir que procesar varios minutos de millones de canciones es muy caro computacionalmente.

2.4.3 El problema del arranque en frío

El término proviene de los automóviles. Cuando hace mucho frío, el motor tiene problemas para arrancar, pero luego de unos minutos llega a una temperatura óptima y funciona perfectamente. Con los motores de recomendación sucede algo parecido, el "arranque en frío" significa que no se han dado las condiciones para que el sistema proporcione buenos resultados.

Tenemos dos categorías de arranque en frío:

1. **Usuarios**: Cuando se registra un nuevo usuario no conocemos sus gustos y preferencias debido a la falta de interacción con la plataforma.

2. **Ítems**: Al lanzar una nueva canción, no tendrá historial de reproducciones y mucho menos valoraciones. Adicionalmente, sucede lo mismo con ítems que solo le interesan a un nicho y por lo tanto tienen datos de uso escasos.

Notar que aunque no tengamos información colaborativa de la canción nueva o de los usuarios en relación a ésta, sí tenemos metadatos y el audio. Acá entra al rescate el enfoque basado en el contenido. Combinando las características del audio como sus metadatos, podemos predecir si le gustará a otros usuarios con los mismos gustos.

2.4.4 Brecha semántica

Existe una gran brecha semántica entre las características de una canción que moldean las preferencias de usuario y su correspondiente señal de audio. El género, *mood*, instrumentación y letras; son propiedades de alto nivel que requieren potentes modelos computacionales para capturar estructuras jerárquicas complejas. Además, algunas propiedades no pueden ser extraídas a partir del audio como por ejemplo: La popularidad del artista, ubicación geográfica o año de publicación. Es por esto que los enfoques colaborativos

suelen funcionar mejor que los en contenido siempre y cuando existan datos de uso.

En este trabajo abordaremos este problema utilizando dos enfoques: Uno tradicional de MIR que utiliza una *bag of words* para representar las canciones, y el otro, redes neuronales convolutivas para extraer features temporales de más alto nivel.

2.4.5 Conjunto de datos

Tradicionalmente, el área de investigación *Music Information Retrieval* estaba reservada solo para la industria ya que no había un dataset relevante y lo suficientemente grande con el cual hacer experimentos de calidad. Es por esto que en el año 2010, investigadores de *LabROSA* Columbia University e integrantes de *The Echo Nest* (adquirida por *Spotify*) crearon el Million Song Dataset (Bertin-Mahieux et al.). El MSD está compuesto por un millón de canciones comerciales populares junto a audio features pre-computados y metadatos. También podemos encontrar otros datasets ligados al MSD, covers de canciones, letras, tags, géneros y registro de datos de uso.

Nuestros experimentos hacen uso de dos de los datasets ligados al MSD:

- *The Echo Nest Taste Profile Subset (“The Echo Nest Taste Profile Subset | Million Song Dataset”)*: Un historial de datos de uso con más de 1 millón de usuario únicos, 300.000 canciones del MSD y 48 millones de registros usuario / canción / cantidad de reproducciones. Éste dataset fue usado en el concurso The Million Song Dataset Challenge (McFee, Bertin-Mahieux, et al.) organizado por *Kaggle*.

- *Tagtraum Genre Annotations* (Schreiber): Clasificación de 15 géneros de 191,401 canciones del MSD.

El problema del MSD es que no tiene audios de las canciones por razones de copyright. Si bien cuenta con features pre-computados, éstos no han sido

documentados correctamente. No nos queda opción que conseguir los audios en crudo de cada canción. Afortunadamente, conseguimos el 99% de los audios del millón de canciones, aproximadamente 650 GB de audios en mp3.

	Fran	**Diego**	**Franco**
Life in Technicolor	**15**	**1**	**10**
Without Me	**10**		**2**
I Gotta Feeling	**1**	**5**	
Paradise city		**15**	**10**

Tabla: Ejemplo de una matriz de cantidad de reproducciones similar al Taste Profile Subset

Para agilizar el entrenamiento, elegimos un subconjunto del MSD. Son 5.000 canciones con audios, un total de 2.9 GB. A su vez, filtramos el Taste profile subset con estas canciones y los 20.000 usuarios con más reproducciones.

3. Representación vectorial de las canciones

En este capítulo, nuestro objetivo es obtener representaciones vectoriales compactas de las canciones que obtuvimos a partir del historial de reproducciones del *Taste Profile Subset*. Veremos el algoritmo WMF, que tomará esa gran matriz y la reducirá en una matriz compacta de vectores de factores o características latentes de las canciones. A partir de esto podremos hacer una recomendación teniendo en cuenta la similaridad del coseno.

Actualmente, Spotify y Netflix por ejemplo, cuentan con un sistema de valoración donde el usuario indica si le gustó o no cierto ítem (canción o película). Nuestro *Taste Profile Subset* es una gran matriz que contiene la cantidad de reproducciones de usuario por canción, lo que es una forma de *implicit feedback.* Así es que podemos inferir que:

- Si un usuario reprodujo muchas veces un ítem, es muy probable que le guste.
- Si un usuario no tiene reproducciones de una canción, no es porque no le guste, si no porque probablemente no la conozca o porque éste cree que no le gustará.

Recordemos que nuestra matriz de reproducciones está formada por 5.000 canciones de los 20.000 usuarios con más actividad. Dado que podemos agrupar usuarios que escuchan frecuentemente las mismas canciones, tendremos una matriz dispersa con muchos ceros. A partir de ésta gran matriz, aplicamos un algoritmo de factorización de matrices para hacerla más compacta. Tradicionalmente, se han utilizado estos algoritmos para predecir *ratings* (de cero a diez por ejemplo), pero como dijimos, nuestros ceros no implican el disgusto de un usuario, sólo que simplemente, no la conoce.

Luego de aplicar el algoritmo obtendremos 2 nuevas matrices: Vectores de canciones y de usuarios. Ahora estos vectores, los de canciones por ejemplo, son los que se llaman **vectores de factores latentes** y dan una representación en un espacio vectorial de baja dimensión de los ítems.

3.1 Weighted matrix factorization (WMF)

El algoritmo WMF, propuesto por Hu et al, aprende representaciones de factores latentes de todos los usuarios e ítems del *Taste Profile Subset*. Éste algoritmo de factorización de matrices modificado apunta a los *implicit feedback datasets (Hu, Koren, and Volinsky).*

Sea r_{ui} la cantidad de veces que un usuario u ha reproducido la canción i. Para cada par usuario-ítem, definimos una variable de preferencia p_{ui} y una variable de confianza c_{ui}

$$p_{ui} = I(r_{ui} > 0)$$
$$c_{ui} = 1 + \alpha log(1 + \epsilon^{-1} r_{ui})$$

donde $I(x)$ es la función indicador, α y ε son hiper parámetros.

La **variable de preferencia** indica si un usuario u ha escuchado alguna vez una canción i. Si el valor es 1, asumimos que el usuario se interesó por la canción. La **variable de confianza** mide la certeza que tenemos sobre ese valor de preferencia. Es una función sobre la cantidad de reproducciones, ya que las canciones con mas reproducciones son las propensas a ser las más preferidas. Si una canción nunca fue escuchada, es el caso donde se dispone de menos información, la variable de preferencia toma el valor cero y la variable de confianza su menor valor: 1.

La función objetivo de WMF está dada por:

$$\min_{x_*,y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

donde λ **es un parámetro de regularización**, x_u es el vector de factor latente del usuario u y y_i es el vector de factor latente de la canción i. Consiste de, un término de *weighted confidence mean squared error* y otro de regularización L2. Notar que la primera suma itera sobre todas los usuarios y todas las canciones, en cambio si usáramos la factorización de matrices sobre *ratings,* descartaría los términos usuario-canción que no tienen *rating*. Volviendo a nuestro caso, tenemos que tomar en cuenta todas las combinaciones. Hu et al. propuso un eficiente método de optimización llamado *alternating least squares (ALS)* que usamos en vez de *Stochastic gradient descent.*

$$\begin{array}{c} \\ U_1 \\ \vdots \\ U_m \end{array} \begin{array}{c} \begin{array}{ccc} I_1 & \cdots & I_n \end{array} \\ \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{bmatrix} \end{array} \sim [U_1 \quad \cdots \quad U_m]^T \cdot \begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} = \mathrm{U}^T \cdot \mathrm{I}$$

Figura:. $U_1 \ldots U_m$ usuarios y $I_1 \ldots I_n$ ítmes. El algoritmo WMF es el encargado de factorizar la matriz de confianza y da como resultado vectores de factores latentes U e I.

3.2 Experimentos

Nuestro subset consta de 5000 canciones y 20.000 usuarios. Aplicamos el algoritmo WMF implementado en la librería de *python Implicit* (Frederickson) con un total de 50 dimensiones. La cantidad de dimensiones es un hiper parámetro que hay que ajustar. Utilizamos 50 porque es un buen comienzo y es sugerido por nuestra bibliografía principal. Obtenemos 2 matrices de factores

latentes, uno de usuarios y otro de canciones. Ahora cada canción está representado como un vector de 50 dimensiones.

La **similitud coseno** es una medida de la similitud existente entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir si ambos vectores apuntan en una misma dirección y sentido. Para cualquier ángulo mayor a cero y menor que 180° entre los vectores, el coseno arrojaría un valor inferior a 1. Si los vectores fueran ortogonales el coseno se anularía, y si apuntan en sentido contrario su valor sería -1. De esta forma, el valor de esta métrica se encuentra en el intervalo cerrado [-1,1].

$$similarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|} \qquad \text{con } A,\ B \text{ vectores.}$$

Dijimos que los vectores de factores latentes representan características ocultas de las canciones. En el primer experimento, junto a los vectores de canciones y su género, haremos una nueva reducción de dimensionalidad con T-SNE en 2D, para graficarlas.

El segundo experimento será: Dada una canción, buscar las 10 canciones más similares sólo con información colaborativa (vectores de factores latentes).

3.3 Resultados

Primer experimento

Podemos establecer una correlación entre la información colaborativa, los artistas y géneros.

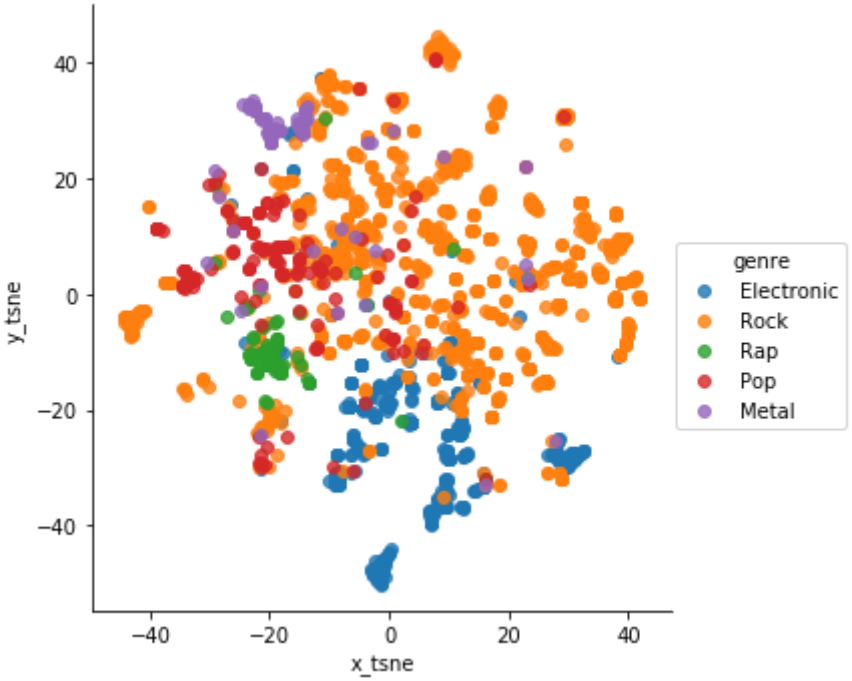

Figura: TSNE de los vectores latentes de 2500 canciones coloreados por género.

T-SNE (Maaten and Hinton) es un algoritmo de *machine learning* para visualización. Es una técnica de reducción de dimensionalidad no lineal adecuado para representar vectores de alta dimensionalidad en un espacio de dos o tres dimensiones.

En cuanto a los géneros, el Rock está bastante disperso seguramente por la enorme cantidad de subgéneros dentro del Rock. Ahora, tanto Electrónica como Rap y Metal están bien agrupados. Con Pop sucede algo similar a Rock pero en menor medida, hay ciertas canciones cercas una de las otras pero otras más dispersas hacia la derecha.

Segundo experimento

Tabla de canción y 10 canciones más similares con su valor de similaridad del coseno.

Consulta	Canciones similares
Green Day **Jesus of Suburbia** Rock	93% - Green Day - St. Jimmy 91% - Green Day - Holiday 90% - Green Day - Give Me Novacaine 88% - Green Day - Whatsername 87% - Green Day - Know Your Enemy 86% - Green Day - She 85% - Green Day - ¡Viva La Gloria! 84% - Green Day - Last Of The American Girls 82% - Green Day - Can You Rock Instructional 80% - Green Day - Burnout
Daft Punk **One more time** Electronic	95% - Daft Punk - Too Long / Steam Machine 93% - Daft Punk - Too Long 93% - Daft Punk - Face To Face 93% - Daft Punk - Voyager 93% - Daft Punk - Face To Face / Short Circuit 92% - Daft Punk - Aerodynamite 92% - Daft Punk - Veridis Quo 92% - Daft Punk - Crescendolls 92% - Daft Punk - Burnin' / Too Long 91% - Daft Punk - Harder Better Faster Stronger
Jack Johnson **Fall Line** Folk	97% - Jack Johnson - Dreams Be Dreams 97% - Jack Johnson - All At Once 97% - Jack Johnson - Cocoon 97% - Jack Johnson - Broken 97% - Jack Johnson - Adrift 97% - Jack Johnson - Cupid 97% - Jack Johnson - What You Thought You Need 97% - Jack Johnson - Same Girl 96% - Jack Johnson - Middle Man 96% - Jack Johnson - Inaudible Melodies
Beyoncé **Halo** R&B	97% - Lady GaGa - Alejandro 97% - Lady GaGa - Just Dance 96% - Miley Cyrus - Party In The U.S.A. 96% - Katy Perry - I Kissed A Girl 95% - Rihanna - Take A Bow 95% - Black Eyed Peas - I Gotta Feeling 95% - Harmonia - Sehr kosmisch 95% - Usher featuring will.i.am - OMG 94% - Justin Timberlake - What Goes Around...Comes Around 94% - Travie McCoy - Billionaire

Nuestro dataset, al estar formado por las 5.000 canciones más escuchadas, contiene más canciones de artistas populares mientras que en los menos populares encontramos muy pocas. Ésto se refleja en la tabla. El artista Coldplay, tiene más de 30 canciones en el dataset, entonces el sistema recomienda solo ese grupo. Esto no es tan bueno, ya que si queremos vender canciones, lo mejor sería tener un poco más de variedad. En las siguientes consultas, los resultados son mucho más variados. El último artista: Beyoncé, tiene sólo 6 canciones en el dataset, pero las canciones que recomendó son relevantes.

En el comportamiento de reproducciones de un conjunto de usuarios podemos establecer que hay grupos de usuarios que escuchan grupos definidos de canciones. Es común que usuarios que disfrutan Pop Electrónico, escuchen de vez en cuando Rock. Pero suponemos que, si un grupo de usuarios escucha música clásica, raramente escuchará Heavy Metal.

4. Enfoque basado en el contenido

Predecir los factores latentes de cierta canción a partir del audio es un **problema de regresión**. Requiere una función que mapee series de tiempo a un vector de números reales. Para alcanzar esto utilizamos dos métodos: El primero sigue un enfoque tradicional en MIR que consiste en extraer features locales del audio y representar cada canción como un vector de *bag of words.* La segunda, consiste en utilizar una red neuronal convolutiva que permite tener en cuenta estructuras temporales de gran escala en sus capas superiores.

Para entrenar los modelos de regresión utilizamos los vectores de factores latentes que fueron obtenidos a partir de WMF sobre los datos de uso.

4.1 Representación Bag of Words

Muchos de los sistemas MIR están basados en éste *feature extraction pipeline* para convertir la señal de audio en representaciones vectoriales para que luego puedan ser usadas en un regresor o clasificador.

1. **Computamos espectrogramas.** A partir de un *sampling rate* de 22050Hz, construimos espectrogramas con ventanas de audio de 1024 cuadros y un solapamiento de 512 muestras; lo que equivale a ventanas de muestreo de 23ms..

2. **Representación MFCC.** En el campo de reconocimiento del habla, se suelen usar 12-14 componentes. Nosotros decidimos usar 13. Entonces, computamos estos 13 componentes a partir de los espectrogramas. También computamos los diferenciales de primero y segundo orden. Ahora cada canción está representada por 39 vectores.

3. **Cuantificar vectorialmente los MFCCs.** Aplicamos el algoritmo de clusterización K-Means sobre todos los vectores con 4000 centroides.

Asignamos todos los vectores MFCCs a su centroide más cercano. Notar que ahora tenemos 4000 features y que cada canción es un subconjunto de éstos.

4. **Agregarlos a una representación Bag of Words.** Contamos la cantidad de veces que aparece cada centroide en una canción. El vector resultante será una representación de *bag of words features* de la canción.

5. **Reducción de dimensionalidad.** A la colección de vectores de features le aplicamos el algoritmo PCA con retención de un 95% de varianza.

Los vectores anteriores representan, una colección de features por canción a partir de la señal de audio. Lo que haremos a continuación será predecir los *latent factor vectors* del capítulo anterior a partir de estos nuevos vectores utilizando técnicas de aprendizaje automático tales como: Regresión lineal, *multilayer perceptron* y una red neuronal densa.

A partir de pruebas con los 3 modelos, vimos que los dos primeros recomendaban mal, casi aleatoriamente. Concluimos que el mejor método que dio resultados fue el de la red neuronal densa.

4.2 Representación con espectrogramas

Nuestro conjunto de datos no sólo contiene metadatos de las canciones, sino archivos de audio en mp3. Los audios constan de 30 segundos y son una muestra para tratar de vender cierta canción, por lo que son de gran calidad.

Para el armado de los espectrogramas, utilizamos la librería para *Python librosa (McFee, Raffel, et al.)*. Es una colección de funciones para procesar señales de audio. Algunas de ellas conocidas, como la STFT mencionada anteriormente o como la función que construye los espectrogramas en la escala de frecuencias de Mel.

Los espectrogramas están construidos a partir de 3 segundos aleatorios del audio original, para agilizar el entrenamiento de los modelos y la conversión de audio crudo a espectrograma. Utilizamos ventanas de 1024 muestras, con una superposición de la mitad (hop-size). Al estar sampleado en 22050 Hz, obtenemos 130 ventanas de 23ms aproximadamente. Luego cambiamos el eje frecuencia de lineal a escala de Mel con 128 componentes para reducir la dimensionalidad. La escala de Mel es logarítmica y es así como el oído humano percibe el volumen.

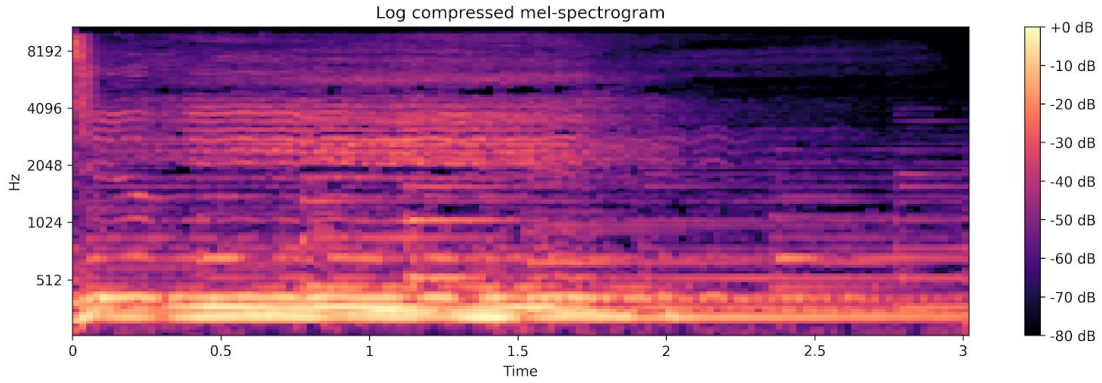

Figura: Fragmento de 3 segundos de la canción Love of my life de Queen.

4.2.1 Redes neuronales convolutivas o CNNs

Las redes neuronales convolutivas fueron un antes y un después en áreas como reconocimiento del habla y visión por computadoras. Este éxito viene dado por tres características:

1. El uso de **rectified linear units** (ReLUs) (Nair and Hinton) lleva a una convergencia más rápida y reduce el problema de *vanishing gradient* (los pesos de las neuronas varían muy poco en relación al error). Éste problema está presente en las redes neuronales tradicionales de muchas capas.

2. El uso de la **paralelización** incrementa la velocidad de entrenamiento, entonces modelos más grandes pueden ser entrenados en una cantidad de tiempo razonable. Utilizamos la librería de *Python Keras* (Chollet) junto a *Tensor Flow (Abadi et al.)*.

3. Una **gran cantidad de datos** es requerida, cuantos más mejor. Cuando los modelos son grandes la cantidad de parámetros es enorme. Es por ello que se necesitan muchos datos. En nuestro caso, el *Million Song Dataset* cumple los requisitos.

Filter shapes

En el campo del procesamiento de imágenes, filtros CNNs cuadrados (de 3x3 o 7x7) son muy comunes. Sin embargo, notar que las dimensiones en imágenes tienen un sentido **espacial**, mientras que en los espectrogramas corresponden a **temporal y frecuencial**. Es por esto que filtros anchos serán capaces de aprender aspectos temporales más largos del audio mientras que los filtros altos serán capaces de aprender características tímbricas más extensas.

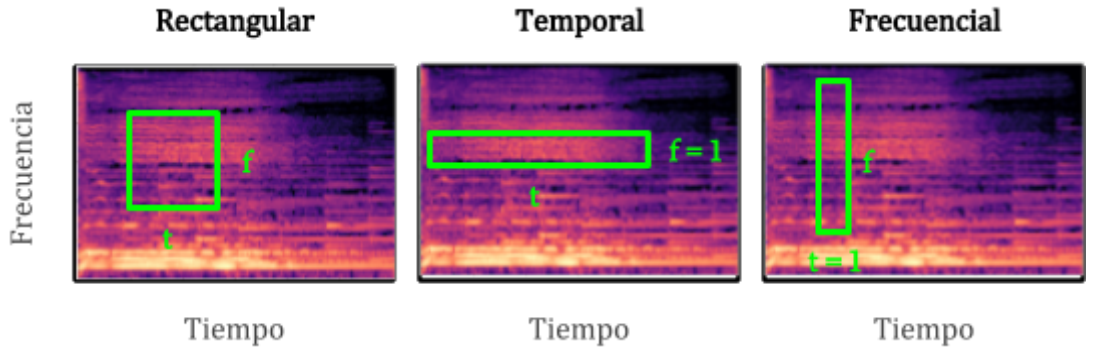

Figura: Los 3 tipos de filtro que vamos a presentar.

1. **Filtros cuadrados o rectangulares:** Son capaces de aprender features de tiempo y frecuencia a la misma vez. Estos filtros pueden aprender diferentes características musicales de acuerdo a su tamaño y forma. Por

ejemplo, el bombo de una batería suena grave (bandas de frecuencias bajas) y en cuanto a tiempo es muy corto. Por lo que un filtro adecuado sería f << F y t << T.

2. **Filtros temporales:** Seteando la dimensión de frecuencia f a 1, estos filtros no serán capaces de aprender features frecuenciales pero sí serán capaces de especializarse en modelar dependencias temporales según el conjunto de datos. Notar que los filtros en sí mismos no pueden aprender características de frecuencias pero las capas más altas sí podrán. Desde una perspectiva musical, estos filtros temporales son capaces de identificar patrones rítmicos y tempo.

3. **Filtros frecuenciales:** Ahora si la dimensión temporal t es 1, los los filtros no podrán aprender aspectos temporales pero sí son capaces de especializarse en modelar características frecuenciales. Parecido a los filtros anteriores, las capas superiores sí serán capaces de tomar en cuenta aspectos de tiempo. Desde el punto de vista musical, se puede identificar aspectos como el timbre y la ecualización de una muestra de audio.

En nuestro caso, estaremos utilizando los últimos filtros. Queremos que nuestro modelo desarrolle características tímbricas para que identifique correctamente los instrumentos utilizados.

Para más información consultar la discusión (Pons).

Arquitectura de la red

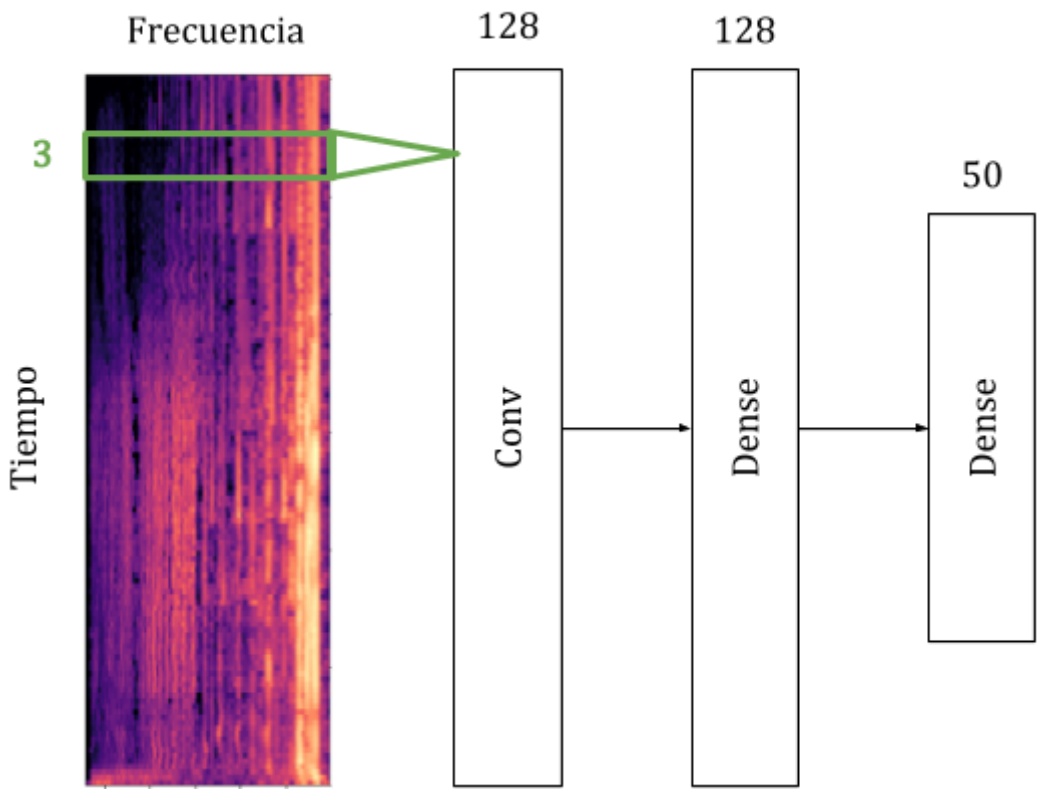

Figura: Arquitectura de la red neuronal convolutiva.

Notar que estas convoluciones son de **una dimensión**, ocurren solo en la dimensión tiempo, no en la dimensión frecuencia. Ésta fue la decisión que explicamos más arriba. La primera capa convolutiva está compuesta de 128 filtros abarcando 3 cuadros de tiempo.

Luego de la capa convolutiva, tenemos operaciones de **max-pooling** para hacer *downsample* de las representaciones y agregar algo de invarianza en el proceso. Se utilizó una ventana de 4 cuadros.

Una capa densa de 128 filtros fue apilada encima de esto y finalmente la capa de salida consta de 50 unidades igualando a la cantidad de dimensiones de los vectores de factores latentes.

Recordar que, para acelerar el entrenamiento, usamos sólo 3 segundos de ventana. Ésto se puede hacer de otras formas: Tomar los 30 segundos y predecir

cada vector latente. O bien, predecir cada ventana consecutiva de 3 segundos por canción y luego promediar los vectores latentes.

Funciones objetivo

Probamos distintas métricas de *loss* a la hora de entrenar. Las más destacadas son: *Mean Squared Error* (MSE), *cosine similarity* (COS) y su versión modificada que comentamos en breve (MCOS). El problema con similaridad del coseno en este caso es que si el modelo llega a mandar a todos los vectores a un mismo punto, la similaridad del coseno es 1. Pero ésto no tiene sentido en el dominio de las canciones, ya que para una canción dada me recomendará con probabilidad 1 todas las canciones.

Con la *Mean Squared Error* tampoco obtuvimos buenos resultados. Entrenamos el modelo durante muchas épocas logrando un error muy pequeño, el problema es que daba muy malas recomendaciones.

Implementamos una nueva métrica, MCOS. Se basa en la similaridad del coseno, pero busca alejar los puntos unos de otros. Cuando las canciones no estén cerca la función da un valor muy bajo. El siguiente algoritmo describe el procedimiento para su cómputo.

Algoritmo para el cómputo de la nueva métrica

Sea y_{true} vectores de factores latentes pertenecientes al dataset de entrenamiento y y_{pred} vectores de factores latentes predecidos por el modelo.

1. Tenemos y_{true}, y_{pred} en $R^{(n \times d)}$ con
   $n = cantidad\ de\ samples,\ d = dimensiones$

2. Obtenemos la matriz de similaridades S en $R^{(n*n)}$ donde:

$$S[i,j] = cosine\_similarity\ (y_{true}[i], y_{pred}[j])$$

3. Sea $S2 = S + 1$

4. La métrica para la k-ésima predicción es:

$$M[k] = \frac{S[k,k]}{\sum S2[:,k]}$$

5. Por último la métrica global será el promedio:

$$\sum \frac{M[k]}{n}$$

4.3 Experimentos

Para comparar realizamos la misma prueba sobre las 4 canciones. Nos aseguramos que éstas canciones pertenecen al *testing set.* Al igual que antes, la métrica que se usó es similaridad del coseno.

Consulta	**Canciones similares** Bag of Words con Red Densa	**Canciones similares** Espectrogramas con CNN
Green Day **Jesus of Suburbia** Rock	Frightened Rabbit - Man/Bag Of Sand Muse - Unnatural Selection Miley Cyrus - East Northumberland High Rammstein - Du Riechst So Gut	Pearl Jam - Last Exit Blink-182 - Violence Slipknot - My Plague Snow Patrol - Hands Open
Daft Punk **One more time** Electronic	Kollaa Kestää - Musti Sotakoira Gwen Stefani - Luxurious Boys Like Girls - Two Is Better Than One Coldplay - What If	Sander Van Doorn - Apple Chromeo - Outta Sight Soda Stereo - Observándonos Jack Johnson - Staple It Together
Jack Johnson **Fall Line** Folk	Third Eye Blind - Company Enya - Storms In Africa Enya - Only Time (Original Version) Lady GaGa / Colby O'Donis - Just Dance	Damien Rice - Amie Bon Iver - The Wolves City And Colour - Body In A Box Led Zeppelin - Ten Years Gone
Beyoncé **Halo** R&B	Drowning Pool - Reason I'm Alive Kansas - Hold On The New Pornographers - The Fake Headlines Robert Johnson - I?'m A Steady Rollin? Man	P!nk - Glitter In The Air Olivia Newton-John - Hochmah Taylor Swift - Tied Together With A Smile OneRepublic - Marchin On

4.4 Resultados

Analicemos primero, los resultados de la CNN con espectrogramas.

- **Jesus of Suburbia de Green Day**: Las 4 canciones que recomienda son de Rock pesado. Se puede escuchar guitarras eléctricas distorsionadas y también una batería ruidosa. Son muy similares a la canción de consulta.

- **En One more time de Daft Punk**, al sistema le tocó una ventana en que sólo aparece un sintetizador o guitarra chillona. La canción Apple, es de género Electronic House y se puede escuchar otro sintetizador, lo mismo sucede con Outta Sight. Curiosamente recomienda Observándonos de Soda Stereo donde se escucha una guitarra eléctrica aguda. En la de Jack Johnson no pudimos encontrar mucha similaridad.

- **Fall Line de Jack Johnson**: Las tres primeras devoluciones son del mismo género y podemos escuchar guitarras electroacústicas y una voz masculina similar a la del cantante. La última de Led Zepellin, es bastante más distinta lo cual es curioso.

- **Halo de Beyoncé**: En esta hay más variedad. Tenemos a Pink y Taylor Swift con voces femeninas y de pop similar a Beyoncé. Marching on de OneRepublic, sigue siendo pop pero sonoramente no es muy similar. Recordemos que los vectores de factores latentes tienen en cuenta al comportamiento de los usuarios, entonces muy probablemente los que disfruten de Beyoncé, disfrutan de Pink, Taylor Swift y One Republic. Ahora Hochmah de Olivia Newton es muy distinta. Se pueden escuchar coros de iglesia y un instrumento de viento similar al sicu.

Ahora nos detengamos a analizar lo que dió el modelo de Bag of Words:

- **Jesus of suburbia de Green Day:** Rammstein y Muse, pertenecen a un Rock pesado, parecido a Green Day. Las otras dos canciones no son relevantes.
- **One more time de Daft Punk:** Sonoramente, ninguna de las 4 es similar. En el fragmento de Coldplay hay algunos agudos que puede estar comparando.
- **Fall Line de Jack Johnson**: No da buenos resultados, Enya es un artista de música celta y espiritual, nada que ver con Jack Johnson. El resto tampoco son parecidas.
- **Halo de Beyoncé**: No da buenos resultados. Las 4 que recomienda son de Rock y la última del clásico Rock and Roll.

Como dijimos antes, los espectrogramas de mel tienen mucha más información que los MFCC. Los resultados de la CNN son muy interesantes, dan la variedad que buscábamos pero sin perder en cuenta la similaridad sonora de las canciones.

En cuanto al enfoque Bag of Words, los resultados son bastante malos. La pérdida de información, al parecer es bastante grande. En las recomendaciones se ve mucha variedad, pero de distinto género y sonoridad. No es lo que estamos buscando.

Probamos consultas con algunos artistas argentinos, el problema es que al dataset lo elegimos con las 5.000 canciones más populares en todo el mundo, entonces hay muy poca información para una canción de Soda Stereo por ejemplo.

5. Clasificación de género

Un género musical es una categoría que reúne composiciones musicales que comparten criterios de afinidad tales como su función (música de danza, religiosa, cine), su instrumentación (vocal, instrumental, electrónica), su contexto social o el contenido de su letra.

La tarea de clasificación de género consiste en, dada una canción (audio o vector de factores latentes) predecir la categoría de género. En esta sección abordaremos canciones basadas en audio. Un sistema de clasificación, podría utilizarse para afinar el problema de *cold start*. Si una persona sube a *YouTube* un cover de violín de una canción de Rock, se podrá predecir su género y que esto ayude a una futura recomendación de otro usuario con gustos similares.

A la hora de clasificar por género, nos encontramos con varios problemas:

1. Categorización amplia: Hay casos donde una canción pertenece a varios géneros. Cada género puede a su vez, tener muchos subgéneros. Por ejemplo el Rock puede subdividirse en: alternativo, folk, indie, hard, punk, entre otros.

2. A nivel audio, la variación de género en una misma canción: Por ejemplo la canción Faith de George Michael. Comienza con una sección de órgano de iglesia, lo cual podría confundirse con una género religioso o clásico. El resto de la canción, es claro que pertenece a una mezcla de Pop, Rock, Funk, entre otros.

3. Hay una brecha semántica importante entre la señal de audio y el género.

Nuestro enfoque será basado en el contenido. Utilizaremos un *subset* del *dataset* mencionado en la sección preliminares, el GTZAN. Es una colección de 500 audios de canciones de 30 segundos clasificados en 5 géneros: Metal, classical, hip hop, country, reggae. Tomaremos 9 segundos aleatorios de cada canción, resultando en 388 cuadros de tiempo.

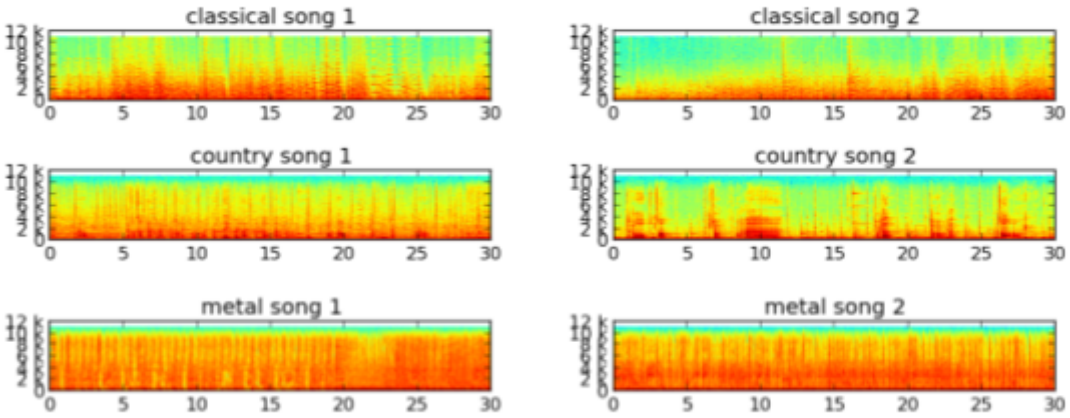

Figura: Ejemplo de dos canciones de los géneros: Classical, country, metal.

5.1 Pipeline

1. Convertimos los audios a espectrogramas de mel: 500x388x128 (canciones x tiempo x frecuencia)
2. Dividimos y mezclamos los audios en *train, test, validation* (75%, 15%, 10%).
3. Entrenamos una CNN con el *training set* usando el *validation set.*
4. Medimos la precisión con el *testing set.*
5. Presentamos los resultados.

La CNN consta de 2 capas convolutivas intermedias de 128 filtros cada una con un *pooling size* de 3 y *dropout* para agregar variación a los datos y prevenir *overfitting*. Encima de esto, apilamos una red densa de 128 neuronas completamente conectada. La última capa, será una densa de 5 neuronas con la función de activación *softmax* que representan los 5 géneros a predecir. La red fue compilada con SGD como optimizador y categorical cross entropy como función de *loss.*

5.2 Resultados

La red fue entrenada durante 27 épocas, dando una precisión del 80% sobre el *training set,* 57% sobre el *validation set* y 70% sobre el *testing set.*

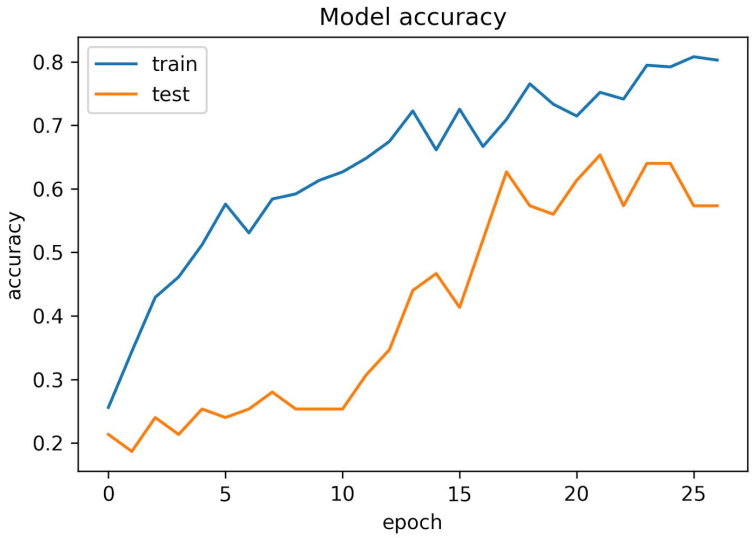

Figura: Precisión del training y testing set durante el entrenamiento.

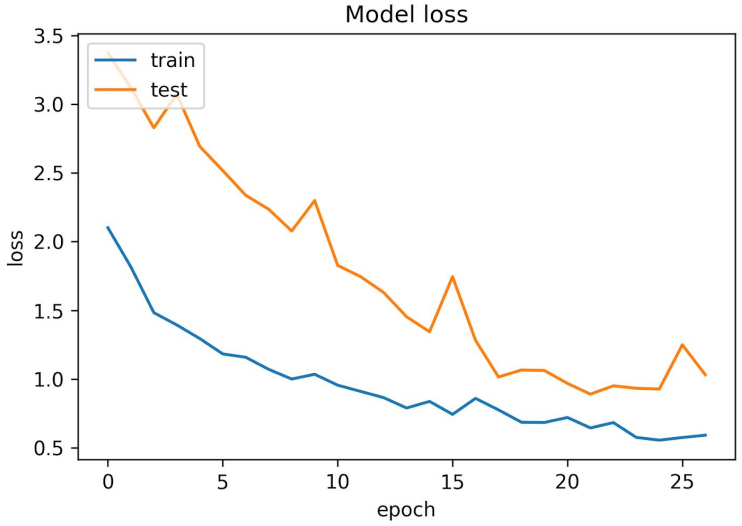

Figura: Valor de categorical cross entropy durante el entrenamiento de la red.

Una **matriz de confusión** es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje automático. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

En nuestro caso podemos ver que para los géneros: Country, metal y classical, el sistema clasifica muy bien. Ahora, el sistema confunde bastante el género reggae con hip hop, logrando una precisión más reducida del 56% y 57% respectivamente. Por último, se puede ver algo de confusión entre classical y country lo cual es curioso.

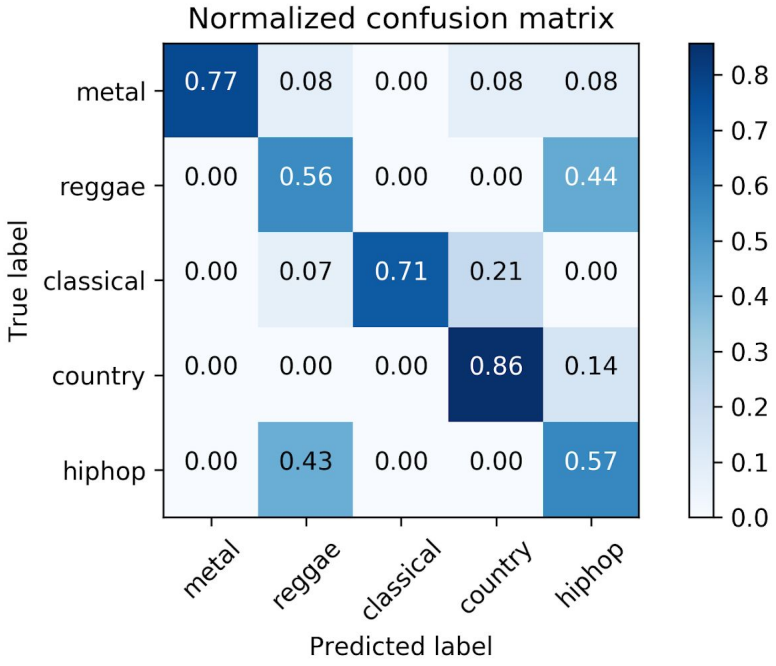

Figura: Matriz de confusión de los géneros predecidos.

6. Conclusión

La recomendación basada en contenido debe usarse para complementarse con el enfoque colaborativo. Vimos que da resultados interesantes y ayuda al usuario a conocer otros artistas y géneros. Aplicamos redes neuronales convolutivas para predecir los vectores de factores latentes de las canciones sin datos colaborativos. Las recomendaciones que hace el sistema son variadas en general, pero son muy relevantes. Podemos concluir que predecir vectores de factores latentes a partir de la señal de audio es un método viable para recomendar música nueva y no popular.

7. Trabajo futuro

El pipeline del sistema presentado está bien definido. Para seguir mejorando los resultados hay mucho por hacer. En esta sección describiré algunas mejoras a realizar.

7.1 Correr los experimentos en un dataset más grande

El *Million Song Dataset*, como su nombre lo indica, está compuesto por un millón de canciones. A partir de ésto pudimos conseguir el 99% de audios de 30 segundos de cada una. Si quisiéramos correr el sistema sobre el dataset completo, tendríamos un problema de big data. Tendríamos que entrenar si o si con GPU y adaptar los algoritmos a una forma más eficiente.

7.2 Tomar ventanas sucesivas de audio

El sistema actual toma una ventana 3 segundos, localizada en forma aleatoria. El problema de ésto es que esta ventana puede caer justo en una parte de la canción no representativa. Por ejemplo la canción *November Rain* de *Guns N' Roses*, comienza con una sección de piano y orquesta lo que puede ser confundido con música clásica. Una opción sería tomar las 10 ventanas de 3 segundos y luego promediar los vectores latentes predecidos para lograr un resultado más representativo y preciso.

7.3 Tomar ventanas de audio más grandes

Computar el espectrograma de una ventana de 30 segundos de audio lleva aproximadamente 4 segundos de ejecución en una *laptop* promedio. Computar el espectrograma de 5000 canciones tomaría alrededor de 5 horas. Ahora haga la

cuenta de cuánto sería en el dataset completo de 1.000.000 de canciones. Muchísimo tiempo. Para una tarea así, habrá que replantear el algoritmo y usar un clúster.

7.4 Transfer learning

El experimento de clasificación de géneros de GTZAN funciona muy bien. El estado del arte es de alrededor un 80% de precisión. Se podría usar las capas intermedias convolutivas ya entrenadas con ese problema y aplicarlo al nuestro.

8. Bibliografía

Abadi, Martín et al. “Tensorflow: A System for Large-Scale Machine Learning.” *OSDI*. Vol. 16. usenix.org, 2016. 265–283. Print.

Bennett, James, Stan Lanning, and Others. “The Netflix Prize.” *Proceedings of KDD Cup and Workshop*. Vol. 2007. New York, NY, USA, 2007. 35. Print.

Bertin-Mahieux, Thierry et al. “The Million Song Dataset.” *Ismir*. Vol. 2. N.p., 2011. 10. Print.

Chollet, François. “Keras: The Python Deep Learning Library.” *Astrophysics Source Code Library* 1 June 2018. Web.

Frederickson, Ben. “Fast Python Collaborative Filtering for Implicit datasets.(2017).” 2017: n. pag. Print.

Hu, Yifan, Yehuda Koren, and Chris Volinsky. “Collaborative Filtering for Implicit Feedback Datasets.” *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*. Ieee, 2008. 263–272. Print.

John, Joyce. “Pandora and the Music Genome Project.” *Scientific Computing World* 23.10 (2006): 40–41. Print.

Kim, Ji-Sung. “Deep Learning Driven Jazz Generation.” *deepjazz.io*. N.p., n.d. Web. 12 July 2018.

Liang, F. “Bachbot: Automatic Composition in the Style of Bach Chorales.” (2016): n. pag. Web.

Logan, Beth, and Others. “Mel Frequency Cepstral Coefficients for Music Modeling.” *ISMIR*. Vol. 270. musicweb.ucsd.edu, 2000. 1–11. Print.

Maaten, Laurens van der, and Geoffrey Hinton. “Visualizing Data Using T-SNE.” *Journal of machine learning research: JMLR* 9.Nov (2008): 2579–2605. Print.

McFee, Brian, Colin Raffel, et al. “Librosa: Audio and Music Signal Analysis in Python.”

Proceedings of the 14th Python in Science Conference. N.p., 2015. 18–25. Print.

McFee, Brian, Thierry Bertin-Mahieux, et al. "The Million Song Dataset Challenge." *Proceedings of the 21st International Conference on World Wide Web*. New York, NY, USA: ACM, 2012. 909–916. Print. WWW '12 Companion.

Nair, Vinod, and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. cs.toronto.edu, 2010. 807–814. Print.

Pons, Jordi. "CNN Filter Shapes Discussion for Music Spectrograms | Jordi Pons." N.p., n.d. Web. 12 July 2018.

Schreiber, Hendrik. "Improving Genre Annotations for the Million Song Dataset." *ISMIR*. ismir2015.uma.es, 2015. 241–247. Print.

Simon, Ian, and Sageev Oore. "Performance Rnn: Generating Music with Expressive Timing and Dynamics." 2017. Web.

Sturm, Bob L. "An Analysis of the GTZAN Music Genre Dataset." *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies*. New York, NY, USA: ACM, 2012. 7–12. Print. MIRUM '12.

"The Echo Nest Taste Profile Subset | Million Song Dataset." N.p., n.d. Web. 12 July 2018.

Van Den Oord, Aäron et al. "Wavenet: A Generative Model for Raw Audio." *CoRR abs/1609. 03499* (2016): n. pag. Web.

van den Oord, Aaron, Sander Dieleman, and Benjamin Schrauwen. "Deep Content-Based Music Recommendation." *Advances in Neural Information Processing Systems 26*. Ed. C. J. C. Burges et al. Curran Associates, Inc., 2013. 2643–2651. Print.