ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Reconocimiento de patrones rítmicos en señales de audio

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Bernardo Marenco

en cumplimiento parcial de los requerimientos para la obtención del título de Magister en Ingeniería Matemática.

Directores de Tesis

Martín Rocamora ..... Universidad de la República
Paola Bermolen ..... Universidad de la República

Tribunal

Matías Carrasco ..... Universidad de la República
Marcelo Fiori ..... Universidad de la República
Luiz Wagner Pereira Biscainho ..... Universidade Federal do Rio de Janeiro

Director Académico

Paola Bermolen ..... Universidad de la República

Montevideo
viernes 1 noviembre, 2019

Reconocimiento de patrones rítmicos en señales de audio, Bernardo Marenco.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 119 páginas.
Compilada el viernes 1 noviembre, 2019.
http://iie.fing.edu.uy/

Agradecimientos

A Martín y Paola, por el apoyo y la paciencia.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

En este trabajo se presenta una metodología para el reconocimiento automático de patrones rítmicos en señales de audio, usando cadenas ocultas de Markov como herramienta de clasificación. Los experimentos reportados se concentran en el ritmo del candombe, en particular en los patrones rítmicos de los tambores repique y piano.

En el caso del repique, se busca identificar en el audio algunos patrones rítmicos, propuestos por Luis Jure en su trabajo "Principios generativos del toque de repique del candombe" [34]. La implementación de la metodología utiliza audio sintético para el entrenamiento de las cadenas ocultas, y los resultados obtenidos en el reconocimiento son muy buenos si el audio que se quiere clasificar es también sintético, obteniendo más del 90 % de acierto en la clasificación. Si se usa audio sintético para entrenar y grabaciones reales para clasificar, el desempeño cae drásticamente, siendo menor a 10 % en las pruebas realizadas. Se discuten algunas alternativas para mejorar la clasificación en ese caso, una de las cuales es implementada. Aún así, la clasificación de audios reales no mejora demasiado, resultando apenas superior al 10 %.

Para el tambor piano, el problema es identificar en el audio qué compases se corresponden con su patrón más típico (referido usualmente como *base de piano*) y cuáles no (lo que se conoce como *piano repicado*). En este caso, tanto el entrenamiento como la evaluación de desempeño se realizan con grabaciones reales, y en ese caso se logra un buen porcentaje en la clasificación (superior al 80 % en todas las pruebas realizadas).

Palabras clave: *procesamiento de audio; cadenas ocultas de Markov; candombe; MIR; patrones rítmicos.*

Esta página ha sido intencionalmente dejada en blanco.

Abstract

This thesis presents a methodology for automatic recognition of rhythmic patterns from audio signals using hidden Markov models (HMMs) as a classification tool. The reported results deal with *candombe* drumming and focus on the rhythmic patterns of the *repique* and *piano* drums.

For the *repique*, the goal is to identify in the audio some of the rythmic patterns proposed by Luis Jure in its paper 'Principios generativos del toque de repique del candombe' [34]. The implementation uses synthetic audio for HMM training, and recognition results are very good if the audio to be classified is also synthetic, with accuracy being over 90 %. If synthetic audio is used for training and classification is performes with real recordings, performance drops dramatically, being less than 10 % in the tests performed. Some alternatives to improve classification in that situation are discussed, one of which is implemented. Even so, classification of real audios does not improve much, with accuracy being barely over 10 %.

For the *piano* drum, the goal is to identify in the audio which bars correspond to its most typical pattern (usually referred to as *piano base*) and which do not (what is known as *piano repicado*). In this case, both training and performance evaluation are carried out using real recordings, and classification results are very good, obtaining over 80 % accuracy in all tests performed.

Keywords: *audio signal processing; hidden Markov models; candombe; music information retrieval; rythmic patterns.*

Esta página ha sido intencionalmente dejada en blanco.

Prefacio

La investigación que da origen a los resultados presentados en la presente publicación recibió fondos de la Agencia Nacional de Investigación e Innovación bajo el código **POS_NAC_2015_1_110063**.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Capítulo 1

Introducción

El problema de la composición de música a través de métodos formalizables (usualmente denominado *composición algorítmica*) ha sido objeto de estudio desde hace aproximadamente 1000 años [48]. Esto indica que el modelado de los procesos creativos en música es un problema que históricamente ha interesado al ser humano. Una buena comprensión de cómo se llevan a cabo dichos procesos no sólo interesa porque permitiría realizar composiciones de forma automática, sino porque también implicaría un avance en la comprensión del funcionamiento de la mente en los procesos creativos. Además de la composición algorítmica, problemas como la simulación de estilos musicales e incluso de algunos intérpretes [51,53] o la generación automática de improvisaciones [13] también han sido objeto de estudio.

En música, la improvisación puede entenderse como el proceso de manipular deliberadamente algunos de los elementos que componen una pieza musical, como por ejemplo la melodía o la armonía, en el mismo momento en que se ejecuta la pieza, en general sin planificación previa. Usualmente, el estudio de la improvisación como fenómeno musical se centra en derivar las reglas que gobiernan este proceso. Si bien es un fenómeno que se caracteriza por la libertad a la hora de tocar, es claro que dichas reglas existen: en general, no todas las posibles combinaciones de sonidos que el músico pueda elegir son válidas, independientemente de qué instrumento o tipo de música se trate.

Aunque la capacidad de improvisar depende de la habilidad que tenga el músico, no es éste el único factor en juego. Esa habilidad se construye en base a un conocimiento en profundidad del instrumento y del estilo musical que se trate. El estudio y análisis de piezas musicales preexistentes juega un papel preponderante en este proceso. Es por ese motivo que el análisis de improvisaciones ha servido como base para derivar las reglas que supuestamente rigen la improvisación, como por ejemplo en [17, 20, 29].

Es en este contexto que surge este trabajo, intentando contribuir al análisis sistemático de los procesos creativos en música, buscando cuáles son las contribuciones que pueden realizarse desde el campo del procesamiento de señales. Para ello, se tomó como caso de estudio al candombe afro-uruguayo. Esta elección fue motivada por dos aspectos: primero, para continuar una línea de investigación que comenzó con el proyecto de grado en Ingeniería Eléctrica [45], aprovechando el

conocimiento generado y los vínculos existentes con personas relacionadas con el estudio del candombe a nivel académico (como Martín Rocamora y Luis Jure); y segundo, como se señala en [62], porque el estudio cuidadoso de una tradición musical particular fuera del paradigma de música comercial occidental puede contribuir a la construcción de modelos más generales y ricos que los que actualmente dominan la investigación en tecnologías de la información aplicadas a la música.

Así, esta tesis está motivada por el trabajo realizado por Luis Jure en [34], donde se postulan una serie de principios generativos para el toque de repique a partir de un axioma y reglas transformacionales. El axioma intenta caracterizar al patrón propio del repique, mientras que las reglas transformacionales indican algunas de las posibles variaciones que se utilizan para modificar dicho axioma y así obtener nuevos patrones rítmicos.

En este trabajo se realizó una clasificación automática de algunos de los patrones rítmicos propuestos por Jure, tanto del axioma como de algunas de sus variaciones. Esta clasificación se llevó a cabo mediante el procesamiento de señales de audio. A partir del enfoque utilizado para la clasificación de patrones de repique, se implementó también un sistema de reconocimiento del patrón básico del tambor piano.

1.1. El candombe afro-uruguayo

El candombe afro-uruguayo, tal y como su nombre lo indica, es un género musical de origen africano propio del Uruguay. Fue desarrollado en el país a partir de la llegada de los primeros esclavos africanos, en el siglo XVIII [22]. Es uno de los rasgos más característicos de la cultura popular nacional, al punto tal que el ritmo ha sido integrado en distintos grados en varios géneros de la música uruguaya, como el tango o el canto popular, y ha dado lugar al candombe-beat y otras formas musicales posteriores [22, 58].

Tradicionalmente, se identifican en el candombe tres estilos o corrientes tradicionales, cuyos nombres surgen de los barrios o calles a los que están asociados: Barrio Sur (o Cuareim), Palermo (o Ansina) y Cordón (o Gaboto), siendo Cuareim y Ansina los más diferentes, mientras que Gaboto es usualmente considerado una variante de Ansina. Una de las principales diferencias entre los dos primeros es el tempo de las interpretaciones: en Cuareim se prefiere el tempo bajo, mientras que Ansina tiene un estilo más rápido (y en Gaboto aún más).

En el candombe se utilizan tres tambores: el chico, el piano y el repique. El toque de todos los tambores consiste en una sucesión de golpes sobre la lonja o el cuerpo del tambor (comúnmente llamado madera). Los golpes sobre la lonja se caracterizan por ser efectuados con ambas manos, con la peculiaridad de que una de ellas sostiene una baqueta (denominada simplemente palo); mientras que el golpe de madera se realiza únicamente con el palo. Las diferencias entre los tres estilos tradicionales están presentes en el uso y la preeminencia que dan a cada tambor: como señala Luis Ferreira en [22]: “En la Cuerda del barrio Sur los chicos son bien notorios, los pianos son los más sencillos, y los repiques tienen intervenciones más medidas y alternadas. En la Cuerda del barrio Palermo los pianos repican

alternativamente y le dan entrada a los repiques, los cuales intervienen mucho más seguido y con más variabilidad que en la del Sur. El toque de Cordón, con similitudes al de Palermo, suele ser fuerte pero como en ondas de menor a mayor intensidad y viceversa; es la Cuerda con más pianos, repicados enfáticamente".

Cada tambor tiene sus particularidades. El chico es el más pequeño, de sonido más agudo, cuyo principal objetivo es establecer, con un patrón virtualmente inalterable, la pulsación constante sobre la que se construye la estructura métrica. El piano es el de dimensiones más grandes y el del sonido más grave. Si bien su estructura básica está definida, tiene un mayor grado de libertad que el chico para la improvisación. Como hace notar Luis Ferreira: "Los tambores chico y piano son como dos polos de energía complementarios que «arman» la base" del candombe. Por último, el repique se encuentra en un punto intermedio: es en general de un tamaño más grande que el chico pero más pequeño que el piano, y se ubica en el registro medio-agudo. Como se señala en [60], es el responsable de generar interés, sorpresa y variedad musical en el candombe. Existe un patrón propio de repique, del que es común que se realicen variaciones improvisadas. Además, suelen ejecutarse figuras que se alejan bastante de ese patrón, por lo que su toque tiene un alto grado de complejidad. Trabajos de Luis Ferreira [21] y Luis Jure [33–35] han avanzado en el estudio de la improvisación, analizando y pautando grabaciones existentes de toques de repique.

1.2. Principios generativos del toque de repique

Para comprender los principios generativos del toque de repique, en [34] Jure comienza por analizar la estructura métrica del candombe, a la que divide en tres niveles métricos. El nivel más bajo, llamado *tatum*, está dado por el patrón del chico, que se mantiene prácticamente inalterado a lo largo del tiempo. Una notación esquemática de ese patrón se muestra en la figura 1.1, junto con la clave del candombe. Este patrón, fácilmente reconocible para un oyente familiarizado con el ritmo, es el encargado de establecer el segundo nivel rítmico, llamado *tactus*. El *tactus* (también llamado genéricamente *beat*) es el pulso perceptivamente más saliente, el que alguien eligiría para llevar palmas o marcar con el pie cuando escucha la música.[1] En el caso del candombe coincide con el pulso dado por los pies al caminar, ya que los intérpretes caminan con el paso sincronizado mientras tocan. De esta manera, la clave implícitamente permite a los distintos tocadores establecer una base de tiempos común.

A su vez, como se ve en la figura 1.1 la clave introduce un tercer nivel métrico, un nivel por encima del *tactus*, que se denominará *compás*. Un compás está compuesto por 16 *tatums* o 4 *tactus*.

Una vez identificados estos tres niveles métricos, Jure plantea como axioma un patrón rítmico que abarca un compás, que se muestra en notación musical en la figura 1.2, junto con la clave. La figura de negra da el pulso del ritmo, en un

[1] Es por ese motivo que en este trabajo se utilizará el término *pulso* para referirse al intervalo de tiempo comprendido entre dos *tactus* consecutivos.

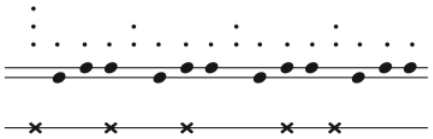


Figura 1.1: Arriba: patrón del tambor chico, junto al nivel métrico introducido por este (*tatum*, puntos simples) y al nivel métrico introducido por la clave (*tactus*, puntos dobles). La línea inferior representa los golpes de mano y la superior los de palo. Abajo: clave del candombe. Figura extraída de [34].

compás de 4/4.

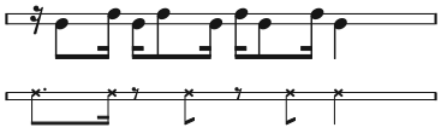


Figura 1.2: Arriba: axioma generativo del toque de repique (nuevamente, la línea inferior representa los golpes de mano y la superior los de palo). Abajo: Clave del candombe. Figura extraída de [34].

Luego, identifica tres niveles constitutivos del axioma, que denomina inicio, núcleo y final. Esta división se muestra gráficamente en la figura 1.3.

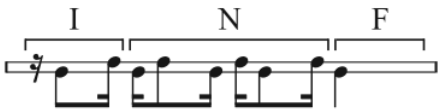


Figura 1.3: División del axioma en sus tres niveles constitutivos: inicio (I), núcleo (N) y final (F). Figura extraída de [34].

A partir de este axioma plantea una primera regla transformacional, que consiste en repetir el núcleo tres veces en lugar de una, como se ve en la figura 1.4. La repetición de tres núcleos no es casual, ya que de esta manera el final F coincide siempre con el cuarto tiempo del compás. El autor llama a esta alteración regla transformacional de expansión, ya que expande el axioma, haciendo que la figura total dure dos compases.

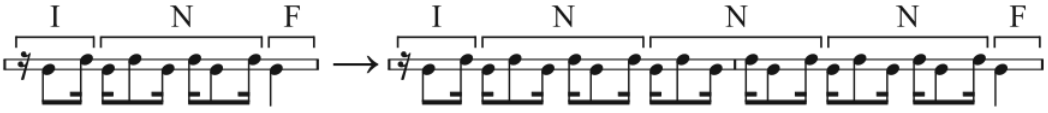


Figura 1.4: Expansión del axioma mediante la repetición de tres núcleos. Figura extraída de [34].

La segunda regla transformacional planteada consiste en la aplicación de la regla de expansión a alguno de los nuevos núcleos que aparecen, dilatando así aún

más la duración total del axioma. En la figura 1.5 se muestra la aplicación de esta regla sucesivas veces. Como se observa, esto causa que siempre haya una cantidad impar de núcleos entre un inicio y un final.

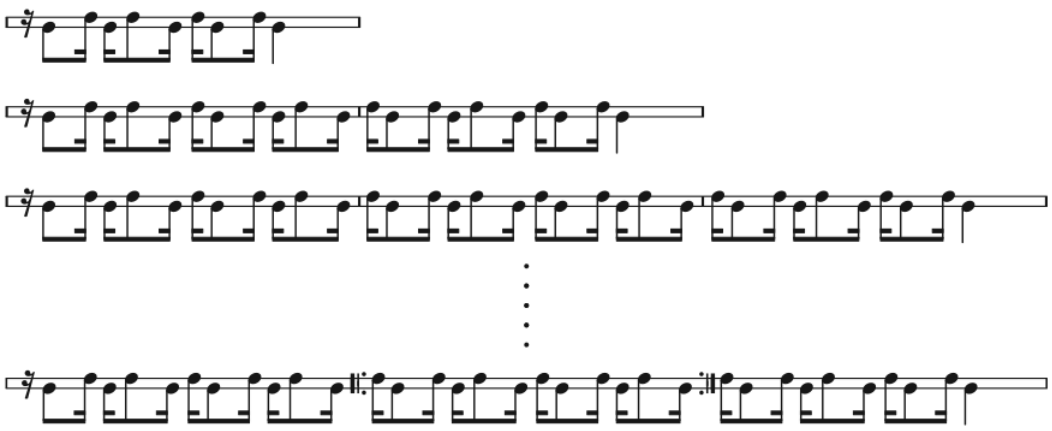


Figura 1.5: Expansión del axioma mediante la aplicación sucesiva de la regla transformacional de expansión. Figura extraída de [34].

El siguiente grupo de transformaciones es llamado por Jure "reglas transformacionales de ornamentación". Como su nombre lo indica, son reglas que implican la ornamentación del patrón de repique, mediante el agregado de golpes en algunos de los elementos constitutivos del axioma (en I y en N específicamente). Cabe aclarar que estas ornamentaciones actúan sobre el axioma sin cambiar su estructura rítmica.

El primer adorno de este tipo consiste en el agregado de un golpe de palo en el tercer *tatum* del patrón I, como en la figura 1.6. Dado que esta ornamentación implica aumentar la cantidad de golpes en ese pulso, Jure se refiere a esta regla como una "densificación" del inicio. Vale destacar que con esta transformación el inicio pasa a ser igual al patrón básico de chico.

Figura 1.6: Densificación del inicio del axioma mediante el agregado de un golpe de palo en el tercer *tatum*. Figura extraída de [34].

Las siguientes ornamentaciones que se consideraron en este trabajo aplican sobre el núcleo N, y también son "densificaciones", es decir, agregan golpes a ese patrón. Ellas son: el agregado de un golpe de palo en el tercer tiempo del primer pulso del núcleo (figura 1.7a), y el agregado de un golpe de mano en el tercer tiempo del segundo pulso (figura 1.7b).

El último grupo de transformaciones es llamado por Jure "de sustitución" pues justamente sustituyen elementos del axioma por otros. Por ejemplo, plantea que

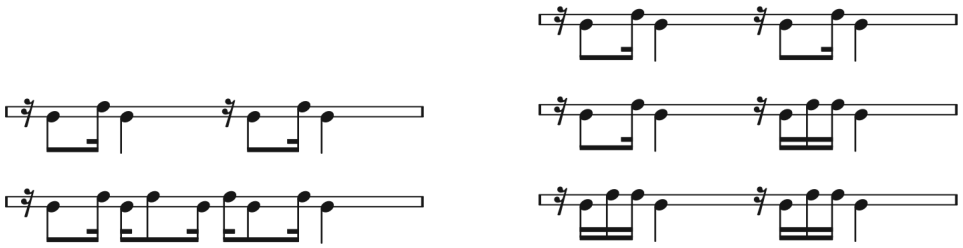
(a) Agregado de un golpe de palo en el tercer tiempo del primer pulso.

(b) Agregado de un golpe de mano en el tercer tiempo del segundo pulso.

Figura 1.7: Densificaciones del núcleo mediante el agregado de golpes. Figuras extraídas de [34].

en secuencias I-N-F (como por ejemplo el axioma sin expandir), el núcleo puede sustituirse por una secuencia F-I, como en la figura 1.8. Es interesante notar que esta sustitución deriva en un compás que es un subconjunto del axioma, como se ve en la figura 1.9a. Además, la sustitución puede combinarse con la regla de ornamentación de I vista anteriormente, para obtener compases como los de la figura 1.9b.

Figura 1.8: Sustitución, en el axioma, de el núcleo por una secuencia **F-I**. Figura extraída de [34].



(a) (b)

Figura 1.9: Primera regla de sustitución: vista como subconjunto del axioma (izquierda), combinada con el adorno de I (derecha). Figuras extraídas de [34].

La última regla de sustitución que se analizó en este trabajo es la sustitución del núcleo N por dos repeticiones de I, como en la figura 1.10. Jure postula que esta sustitución solo puede hacerse cuando N sigue a I, como en el comienzo del axioma.

Debido a que la sucesión I-I toma el lugar de N, hereda su regla de expansión, como se muestra en la figura 1.11. Además, la sustitución puede combinarse con la ornamentación de I vista anteriormente, como en la figura 1.12 (que a su vez puede expandirse como antes).

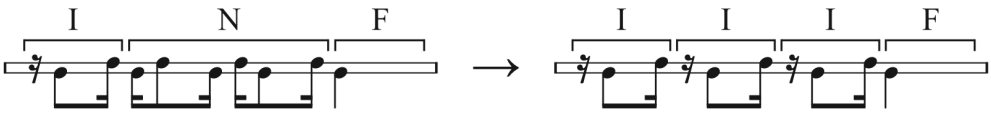


Figura 1.10: Regla de sustitución de un N por una sucesión I-I. Figura extrída de [34].

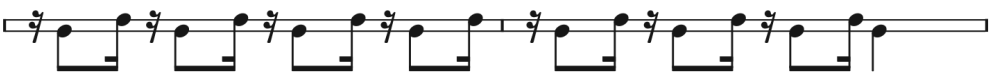


Figura 1.11: Expansión de la sucesión I-I, que ocupa el lugar del núcleo N. Figura extraída de [34].

Figura 1.12: Sustitución de un N por una sucesión I-I adornada. Figura extraída de [34].

Así, el sistema propuesto en este trabajo para reconocer patrones rítmicos de repique tiene como objetivo reconocer aquellos que surjan de la combinación de las distintas reglas aquí expuestas.

1.3. Motivación y enfoque

Las cadenas ocultas de Markov (en inglés, *Hidden Markov Models* o HMMs) han sido muy utilizadas en campos tan diversos como las telecomunicaciones [38], la medicina [9] o la computación [54]. En particular, han sido aplicadas exitosamente en problemas de procesamiento de voz hablada [28, 56], sobre todo en reconocimiento de palabras. Usualmente, el enfoque utilizado consiste en segmentar temporalmente las palabras y reconocerlas utilizando cadenas ocultas de Markov.[2] La segmentación en general es bastante directa, sobre todo en el caso de reconocimiento de palabras aisladas a partir de grabaciones donde se tiene un único hablante en condiciones de estudio (es decir, el ruido de fondo es prácticamente inexistente): la envolvente de amplitud de la señal cambia bruscamente cuando existe una palabra.

Una vez segmentada la palabra, el reconocimiento se lleva a cabo utilizando HMMs. En general, para reconocer un pequeño conjunto de palabras, se entrena una cadena por cada palabra. Sin embargo, si el corpus de palabras es grande, se estila realizar el reconocimiento en sub-unidades de la misma, usualmente fonemas o morfemas. El uso de fonemas es menos común, dado que cuanto menos compleja sea la unidad a reconocer, hay más variabilidad en su estructura dentro del discurso

[2]Por más información sobre cómo utilizar cadenas ocultas como herramienta de clasificación, ver el Capítulo 2.

hablado. Por otro lado, el reconocimiento de palabras enteras requeriría entrenar demasiadas cadenas, sobre todo si el vocabulario es grande. Una solución intermedia es la utilización de morfemas: son lo suficientemente complejos para tener una estructura determinada y cuasi-invariante entre hablantes, pero la cantidad de morfemas que existen es relativamente acotada. Así, en el ejemplo de procesamiento de lenguaje hablado, para reconocer la palabra "panadería", se entrenarían tres cadenas para reconocer los tres morfemas "pan", "ad" y "ería". Esto permite además la reutilización de cadenas para el reconocimiento de distintas palabras: la cadena entrenada para reconocer "pan" también se utilizará para detectar la palabra "**pan**demonio" o "atra**pan**te".

A partir de estos ejemplos empieza a surgir el paralelismo con el problema que concierne a esta tesis: una ejecución de repique que siga las reglas de Jure podría pensarse como un "discurso", donde cada "palabra" esté formada por la concatenación de"morfemas" del tipo I, N o F.[3] Un pasaje del libro "Los tambores del candombe" de Luis Ferreira es especialmente interesante aquí, pues inadvertidamente muestra esta visión del candombe como una "conversación" entre tambores: "El tambor repique regula la energía total: entre la expansión (con el toque repicado) y la contención (con el toque madera); el repique llama a subir y llama a apurar para incrementar, respectivamente, la intensidad y la velocidad de ejecución de la música; el repique responde a los repicados o toques llamadores de los pianos, y conversa con los demás tambores repique" [22].

Así, siguiendo este paralelismo, si se quieren identificar los elementos que componen esa "conversación" de repiques podría pensarse en entrenar tres HMMs, una por cada elemento constitutivo del axioma, para reconocer esos patrones rítmicos y lograr decodificar qué fue lo que "dijo" el intérprete, es decir, reconocer a partir del audio dónde se ubican las células I, N y F, y cómo fueron combinadas para lograr una interpretación musicalmente coherente.

Esencialmente ese fue el camino que se siguió en este trabajo: intentar generar un sistema de reconocimiento de las partes constitutivas del axioma de Jure, para poder decodificar esos "discursos" musicales. Si continuamos con el paralelismo de reconocimiento de voz, el paso cero de un sistema con ese fin sería el de segmentar las unidades a reconocer, es decir, dividir el audio en segmentos que sean susceptibles de ser reconocidos como I, N o F. El problema que surge es que, a diferencia del caso de voz hablada, el "discurso" musical no tiene descansos entre "palabras", si no que es un discurso continuo donde el reconocimiento y la segmentación son problemas íntimamente ligados. En este trabajo, el camino que se tomó para evitar este problema fue muy simple: si se divide el núcleo del axioma en dos como en la figura 1.13, se tienen cuatro morfemas (I, N_1, N_2 y F) que tienen igual duración: un pulso, es decir, el intervalo de tiempo entre dos *beats* consecutivos.

Por lo tanto, si lo que se quiere reconocer son estas cuatro unidades, la segmentación del audio se soluciona si se conoce la ubicación temporal de los *beats* de la pieza. Si bien no es simple, esta tarea es una de las aplicaciones más clásicas del campo conocido como *Music Information Retrieval/Research* (o *MIR*),

[3]No en vano un conjunto de reglas como las de Jure se conocen en este contexto como una "gramática".

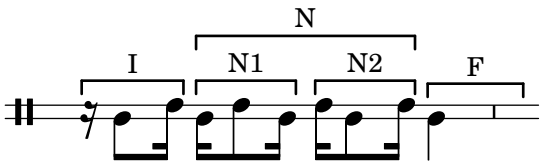


Figura 1.13: División del núcleo del axioma en dos partes de un pulso de duración.

por lo que existe extensa literatura al respecto [16, 19, 39, 50]. Además, existen algoritmos automáticos que desarrollan esa tarea específicamente para candombe afro-urugayo [49,58], con muy buen desempeño. Así, la división del audio en pulsos no es un escollo si se toma este enfoque; se asume que los tiempos de *beats* pueden determinarse, por lo que no se presenta aquí un trabajo adicional en esa dirección.

El primer camino tomado en este trabajo fue entonces intentar reconocer esas cuatro unidades rítmicas usando HMMs. Para ello, se entrenará una cadena para que reconozca cada unidad, y, una vez entrenadas, esas cuatro cadenas se utilizarán para reconocer nuevos pulsos; en el capítulo 3 se detalla cómo se realiza tanto el entrenamiento como la clasificación.

El principal problema con este camino es que, si el reconocimiento de pulsos se hace de forma aislada (es decir, cada pulso de la pieza se clasifica de forma independiente del resto), las reglas gramaticales de Jure no tienen por qué respetarse: nada del esquema de clasificación planteado hasta ahora restringe a que, si un pulso es reconocido como un I, el pulso siguiente deba ser un I, un N_1, o un F, pero no un N_2 como las reglas de Jure establecen. En el equivalente de procesamiento de voz, el esquema de clasificación a nivel de morfemas conlleva necesariamente un procesamiento posterior conocido como "decodificación léxica": se evalúa si la secuencia de morfemas es efectivamente una palabra válida del lenguaje (lo que equivaldría aquí a imponer que la secuencia de pulsos clasificados respete las reglas de Jure que establecen en qué orden pueden aparecer los patrones I, N y F). Además, si el problema de voz no es solo reconocer palabras, si no que se busca reconocer frases a partir de los morfemas, en general se impone otra etapa, llamada de "análisis sintáctico": una vez reconocida la secuencia de morfemas como una palabra válida, se establece si la secuencia de palabras forma una frase sintácticamente coherente.[4]

En este trabajo se exploró un camino que permite imponer ciertas restricciones léxicas al reconocimiento, pero que no cambia demasiado el esquema de clasificación ni implica agregar etapas de procesamiento posteriores. Ese camino consiste en reconocer ya no elementos que sean de un pulso de duración, si no de un compás (es decir, reconocer a nivel de compases en lugar de a nivel de pulsos). En el equivalente de voz, esto sería como pasar de un sistema de reconocimiento a nivel de morfemas a uno de reconocimiento a nivel de palabras.

Por lo tanto, en esta otra opcion se planteó la división de la pieza en compases y la clasificación se realizó para patrones que duren un compás, según las siguientes

[4]Por más información sobre los procedimientos de decodificación léxica y de análisis sintáctico, ver los capitulos VI. "Implementation of Speech Recognizers Using HMMs" y VII. "Connected Word Recognition using HMMs" de [56].

definiciones:

- **Axioma** (A): consiste en la aparición, en un mismo compás, de los tres niveles constitutivos del axioma. Dicho de otra manera, el patrón A consiste en la ejecución, sin ninguna transformación, del axioma original planteado por Jure, es decir A = I-N_1-N_2-F. Este patrón se corresponde con el compás de la figura 1.14.

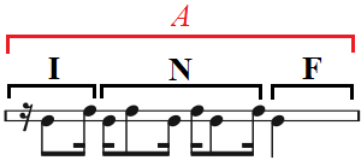


Figura 1.14: Ejemplo de patrón tipo *A*.

- **Comienzo** (C): es el comienzo de la aplicación de la regla de expansión. Esto implica que el patrón C está compuesto por un inicio I, un núcleo entero (es decir, la sucesión N_1-N_2), y la mitad de otro núcleo N (un N_1), que se resuelve en el compás siguiente. Este patrón se corresponde con el primer compás de la figura 1.15.

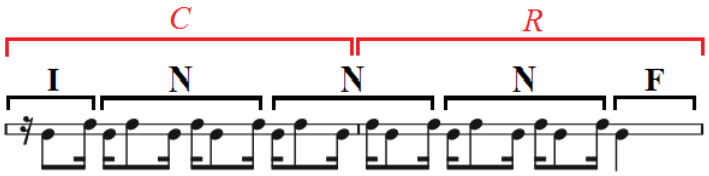


Figura 1.15: Ejemplo de patrones tipo *C* y *R*.

- **Resolución** (R): es la resolución de la regla de expansión, por lo que consiste en la segunda mitad de un núcleo (un pulso N_2), otro núcleo (sucesión) N_1-N_2) y un final F. Este patrón se corresponde con el segundo compás de la figura 1.15.
- **Liso** (L): si en el ejemplo de la figura 1.15 en lugar de resolver la expansión en el segundo compás se hubiese aplicado nuevamente una expansión, el segundo compás resultaría ahora compuesto por medio núcleo proveniente del compás anterior, un núcleo entero, y otro medio núcleo que se traslada al compás siguiente. Ese segundo compás es un ejemplo del patrón L. En la figura 1.16 se ejemplifica esta situación, de donde es claro entonces que L = N_2-N_1-N_2-N_1.
- **Madera** (M): consiste en la realización de la clave dentro de un compás. Si bien este patrón no surge del axioma o de las reglas discutidas anteriormente, fue agregado para enriquecer el modelo, ya que la mayoría de las interpretaciones de repique comienzan por la ejecución de uno o más compases de clave. Un ejemplo de este patrón se muestra en la figura 1.17.

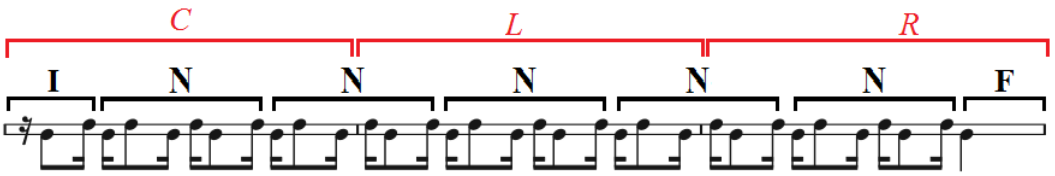


Figura 1.16: Ejemplo de patrones tipo *C*, *L* y *R*.

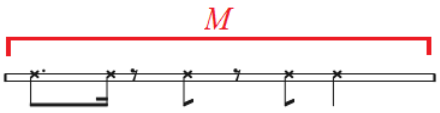


Figura 1.17: Ejemplo de patrón tipo *M*.

La ventaja que tiene este modelo es que ya tiene incorporadas algunas de las restricciones que impone la gramática de Jure, haciendo innecesaria la etapa de decodificación léxica, aunque sí se sigue manteniendo la necesidad de un análisis sintáctico posterior.

Para resumir, se tienen dos propuestas para reconocer patrones rítmicos del toque de repique: una a nivel de pulsos y otra a nivel de compases; en este trabajo se implementaron ambas. En el capítulo 3 se verá que la metodología elegida para el reconocimiento es esencialmente la misma para ambos niveles métricos, mientras que en el capítulo 4 se presentan los resultados obtenidos en cada caso. Además, en la sección que sigue se presenta una propuesta para la clasificación automática del patrón básico del tambor piano, que surgió como continuación natural del enfoque elegido para los patrones de repique.

1.4. El patrón básico de piano

El tambor piano, como se dijo, es el de registro más grave de los tres tambores del candombe. Como señala Jure en [34], es el tambor "de más riqueza y complejidad en su técnica de ejecución, y el que presenta también mayor grado de diferencias entre los distintos estilos barriales". En rasgos generales, como se señala en [58], el toque de piano tiene dos modalidades: un patrón básico (conocido como "piano base") y otras figuraciones más complejas (usualmente englobadas bajo el término "piano repicado"[5]). En la sección 1.1 ya vimos que una de las diferencias entre los distintos estilos barriales consiste en la disposición de los pianos a "repicar": en Barrio Sur los pianos son más sencillos, mientras que en Palermo tienden a las figuraciones complejas (y aún más en Cordón).

En una primera aproximación muy simple, la frase básica de piano se muestra en la figura 1.18. Allí también se muestra la clave; a partir de esa superposición puede verse que, como señala Jure, la base de piano "coincide con la estructura de acentos de la clave" [34]. Así, como la clave, el piano cumple en este caso

[5]Este nombre proviene de su similitud con el patrón básico de repique.

el rol de establecimiento o delineamiento del segundo nivel métrico, el *beat* (e implícitamente, del tercer nivel, el compás). Aquí se comprende más cabalmente la afirmación de Ferreira de que "Los tambores chico y piano son como dos polos de energía complementarios que «arman» la base" del candombe: el chico establece el nivel métrico más bajo, mientras que el piano (y la clave) establecen los dos siguientes.

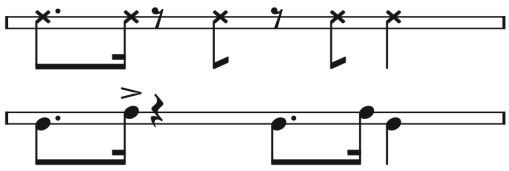


Figura 1.18: Arriba: patrón básico del tambor piano. Abajo: clave del candombe. Figura extraída de [34].

Por otro lado, una aproximación muy básica a un patrón del tipo "piano repicado" se ve en la figura 1.19. Ahí se observa la similitud con el patrón básico de repique, dejando más en claro el por qué de su nombre.

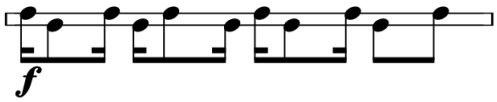


Figura 1.19: Ejemplo de piano repicado. Figura extraída de [58].

Dada la "riqueza y complejidad" de la que habla Jure, no se pretende aquí realizar un análisis extensivo del toque de piano[6]. Simplemente, a los efectos de este trabajo bastará con establecer que existen, a grandes rasgos, dos modalidades del toque de piano: la ejecución de un patrón básico, y la interpretación de otros patrones, que pueden ser más o menos complejos, y que agruparemos bajo el término general "repicados" o "variaciones". El objetivo será entonces reconocer, a partir del audio de una ejecución, cuáles compases corresponden a ejecuciones de la base, y cuáles no pertenecen a ese grupo. La clasificación se realizará usando HMMs, siguiendo la idea de lo expuesto anteriormente para reconocimiento de patrones de repique, es decir, se entrenará una HMM para reconocer cada compás de base. En el capítulo 3 se explica cómo se lleva a cabo esa tarea.

Antes de explicar los detalles de cómo se lleva a cabo la clasificación, en la sección que sigue se hace una breve reseña de algunos trabajos publicados que guardan alguna relación con el trabajo que se desarrolló en esta tesis.

[6]Por una discusión más profunda, que incluye una aproximación computacional al agrupamiento de patrones de piano según estilos barriales, ver [58].

1.5. Trabajo relacionado

Como ya fue mencionado, el uso de HMMs en sistemas de reconocimiento de voz hablada ha sido ampliamente extendido. En particular, y tal vez hasta la reintroducción de las redes neuronales en el área a fines de la primera década del 2000, las cadenas ocultas de Markov han sido la herramienta de referencia en el tema. Trabajos tan influyentes como el fabuloso tutorial de HMMs elaborado por L.B. Rabiner a fines de la década del '80 [56] o el artículo escrito en conjunto con B.H. Juang en 1991 [30] son considerados fundamentales en la disciplina, o al menos en el uso de HMMs como herramienta para reconocimiento de voz hablada. En particular, el tutorial [56] debería ser un punto de referencia para cualquier aplicación que intente utilizar HMMs como herramientas de clasificación (no sólo de audio), por lo claro y lo profundo de su exposición, que va desde los aspectos más básicos de una cadena oculta (qué es, cómo se estiman sus parámetros, cómo se pueden usar como herramientas de clasificación) a los más complejos, llegando al punto de describir un sistema completo de reconocimiento de voz basado en HMMs. El capítulo 2 de esta tesis expone y condensa algunos de los aspectos de ese trabajo, en particular los que explican cómo utilizar HMMs para clasificar secuencias.

El trabajo de Huang también es importante en el tema, pero desde un punto de vista más teórico. En [31, 32] se presentan de manera muy detallada los procedimientos de reestimación de parámetros de una HMM, en el caso de que las observaciones de la misma se asuman como una mezcla de distribuciones multivariadas, extendiendo lo realizado por L. Liporace en [41], que a su vez construía sobre el trabajo realizado por L. Baum para distribuciones unidimensionales en [6–8]. Baum fue quien sentó las bases para el cálculo eficiente de la probabilidad de una secuencia de observaciones emitida por una HMM (el algoritmo *forward-backward*) y de la reestimación de sus parámetros en el proceso de entrenamiento (el algoritmo Baum-Welch). Ambos serán expuestos con más detalle en el capítulo 2. También cabe mencionar en este punto el trabajo de A. Viterbi [63], que introdujo el algoritmo que hoy lleva su nombre y permite estimar la secuencia de estados ocultos visitados por una HMM, a partir de una secuencia de observaciones (que también se expone en detalle en el capítulo 2). Existe además un artículo muy completo de G.D. Forney [25] donde este algoritmo es expuesto con suma claridad.

Finalmente, en el área de HMMs aplicadas a reconocimiento de voz se destaca el trabajo de C.J. Leggetter y P.C. Woddland de 1995 [40], donde se expone el proceso de reestimación de parámetros de una HMM entrenada en ciertas condiciones acústicas para clasificar mejor señales con propiedades acústicas diferentes. Este procedimiento, que en el trabajo Leggetter y Woodland solo implica reestimar las medias de las densidades de observación de la HMM, se utilizará en el capítulo 4 para intentar clasificar interpretaciones reales de repique a partir de un sistema entrenado con audio sintético. Woodland (et. al.) extendió este trabajo en un artículo de 1996 [26], donde se deriva una forma de reestimar también las varianzas de esas densidades. Si bien esa reestimación no fue implementada en este trabajo, es una posibilidad a explorar en trabajos futuros, y puede ser de interés para

personas interesadas en el tema.

Respecto al uso de cadenas de Markov o cadenas ocultas de Markov en aplicaciones relacionadas a la música, uno de los trabajos a destacar es el realizado por C. Ames en 1989 [4], donde se plantea un tutorial para el uso de cadenas de Markov (a secas) para composición algorítmica de música. De manera similar, se pueden mencionar los trabajos de F. Pachet [51–53] en esa línea. En particular, [52] y [53] desarrollan la idea de "cadenas de Markov con restricciones", que implican forzar a una cadena de Markov a que atraviese ciertos estados en tiempos definidos, pasando de una cadena homogénea (una en la que la probabilidad de transición entre estados es independiente del índice temporal) a una no homogénea. De todas maneras, todos estos trabajos plantean el uso de cadenas de Markov como herramientas para composición algorítmica, no como herramientas de clasificación.

Un trabajo que combina ambas perspectivas (síntesis y análisis de música usando cadenas de Markov) es el realizado por M. Farbood y B. Schoner en 2001 [20]. Allí, las cadenas (de nuevo, cadenas a secas, no cadenas ocultas) son utilizadas como modelo composicional, para generar contrapuntos al estilo del compositor italiano del siglo XVI Giovanni Pierluigi da Palestrina. Lo que lo acerca a este trabajo es que los modelos markovianos allí asumidos son también utilizados para analizar las propiedades del contrapunto, como los cambios de cadencia o las relaciones entre las voces del contrapunto (si ambas suben al mismo tiempo, por ejemplo).

Más cercano al trabajo de esta tesis es el realizado por R. Chen et. al. en 2012 [12], en el que se plantea un sistema para el reconocimiento automático de acordes usando cadenas de Markov de duración explícita. Este concepto, muy bien explicado en el tutorial de Rabiner [56], implica introducir una distribución a la probablidad de que la cadena se mantenga en un estado oculto, en lugar de asumirla fija.[7] Dado que en esta tesis se trabaja con instrumentos de percusión, que se caracterizan por no tener información tonal bien definida, el marco en el que los autores de ese artículo trabajan no es de mucha utilidad para este problema en particular, pero por la proximidad con la temática de esta tesis vale la pena mencionarlo.

En el caso específico de instrumentos de percusión, vale la pena destacar los trabajos de P. Chordia con la *tabla*, un instrumento de percusión de la India [13,14]. En particular, su trabajo de 2011 junto con A. Sastry y S. Sentürk [14], utiliza cadenas y cadenas ocultas de Markov, ambas de largo variable, para formar un modelo predictivo de la *tabla*, y utilizan ese modelo para reconocer distintos golpes de tabla a partir de audio. El término largo variable hace referencia a que, en el modelo más básico, las probabilidades de transición de una cadena de Markov solo dependen del estado en tiempo $t-1$ y del estado en tiempo t; esta restricción es

[7] En el caso de una HMM cuya probabilidad de mantenerse en el estado i sea a_{ii}, la probabilidad de que se mantenga en ese estado durante d observaciones es $p_i(d) = a_{ii}^d(1-a_{ii})$. Las HMMs de duración explícita pretenden modificar el marco conceptual de las HMMs para admitir distribuciones $p_i(d)$ cualquiera.

extendida para permitir que la probabilidad dependa también del estado en tiempo $t-2$, y, de forma más general, para todos los estados hasta tiempo $t-n$. Este n se conoce como el "largo" o "orden" de la cadena de Markov. El sistema propuesto por Chordia utiliza varias cadenas de distinto largo en la etapa de reconocimiento (desde orden 0 hasta orden 3). Este enfoque es más complejo que el elegido en esta tesis (en el que se usaron cadenas ocultas de orden 1, es decir, la formulación clásica de Markov); sin embargo, ambos trabajos tienen en común el uso de coeficientes cepstrales de frecuencias mel (MFCCs por sus siglas en inglés) como vector de observaciones de la cadena oculta; en el capítulo 3 se explica detalladamente cómo se utilizan los MFCCs en este caso. Además, en ambos trabajos los estados ocultos se corresponden o están asociados de alguna manera a golpes de percusión; la diferencia es que en el trabajo de Chordia lo que se quiere descubrir es, en cada instante, qué tipo de golpe se realizó, mientras que en esta tesis, lo que se quiere reconocer son patrones rítmicos (y no golpes de manera aislada).

Otro trabajo similar (de hecho es anterior, y Chordia lo referencia constantemente) es el de O. Gillet y G. Richard de 2003 [27]. En él también se trabaja con la *tabla* y, al igual que en el trabajo de Chordia, el objetivo es ubicar y reconocer automáticamente los golpes de *tabla* a partir de una grabación de audio. Para la primera tarea utilizan métodos de detección de *onsets* y de estimación de *tempo*, mientras que para la segunda esa información es combinada con una HMM. Si bien los estados ocultos de la HMM son características derivadas del espectro (como en esta tesis), los autores eligen utilizar una descripción del mismo como una mezcla de 4 distribuciones gaussianas unidimensionales. Así, las características allí usadas son los 8 parámetros de estas gaussianas (media y varianza). Es interesante notar que los autores reportan que probaron utilizar MFCCs como características, pero obtuvieron un peor desempeño.

Ese trabajo además está a medio camino entre el de Chordia y el de esta tesis, en el sentido de que no reconoce golpes aislados, pero tampoco patrones rítmicos. En realidad, los autores optan por reconocer pares de golpes de tabla (es decir, buscan reconocer una sucesión de dos golpes). Esto de alguna manera se acerca un poco más al reconocimiento de patrones rítmicos, aunque no llega a ser exactamente eso. Nuevamente, aquí los estados ocultos de la cadena están asociados a golpes (más precisamente, cada estado está asociado a una posible combinación de dos golpes). El reconocimiento, al igual que en el trabajo de Chordia, y a diferencia de esta tesis, se realiza mediante el algoritmo de Viterbi: una vez "vista" la secuencia de observaciones, se determina con Viterbi la secuencia de estados ocultos (y por lo tanto, la secuencia de golpes). En esta tesis, el reconocimiento se lleva a cabo comparando la probabilidad de la secuencia de observaciones para distintas cadenas, entrenadas cada una para reconocer un patrón rítimico. En la sección 3.2 se explica con más detalle ese proceso.

El último trabajo a mencionar es el de 2009 de J. Paulus y A. Klapuri [55], en el que usan HMMs para ubicar y reconocer automáticamente golpes de batería. Nuevamente, utilizan MFCCs como observaciones de la cadena (no usan solo los MFCCs, si no que computan además sus diferencias de primer y segundo orden). Consideran, al igual que esta tesis, que cada estado oculto está asociado a un tipo

de golpe o a un silencio; la diferencia es que la batería tiene más tipos de golpes posibles que un tambor de candombe, y que los sonidos que produce la batería son más diferentes entre sí. El reconocimiento se realiza, al igual que en los trabajos de Gillet y Chordia, mediante el algoritmo de Viterbi: una vez calculada la secuencia de observaciones (los MFCCs de toda la pieza) se estima la secuencia de estados ocultos visitados. Esto permite a la vez la ubicación temporal y la clasificación de los golpes, ya que los estados ocultos representan golpes en la batería o silencios; así, los golpes se ubicarán en aquellos tiempos en los que la cadena se encuentre en un estado asociado a un golpe. En el momento de la clasificación, los autores realizan además un ajuste de los parámetros de la cadena entrenada, de manera de dar cuenta de las diferencias acústicas entre las señales de entrenamiento y las que se intenta clasificar. Este ajuste es realizado mediante el algoritmo de Leggetter y Woodland que se referenció más arriba.

En general, todos estos trabajos de clasificación de instrumentos de percusión usando HMMs tienen como diferencia fundamental con el expuesto en esta tesis algo que ya fue mencionado varias veces: reconocen golpes aislados, y no frases musicales o patrones rítmicos. Sin embargo, presentan un buen panorama de los problemas del área abordados por la comunidad científica hasta el momento, usando herramientas similares a las aquí propuestas. Cabe mencionar que el trabajo más reciente reseñado es del 2009, ya que a finales de los 2000 las HMMs fueron sustituidas por las redes neuronales como herramientas estándar para reconocimiento en señales de audio.

Con esto finaliza el capítulo introductorio de la tesis. El resto del documento se estructura de la siguiente manera: el capítulo 2 desarrolla los conceptos de cadena de Markov y cadena oculta de Markov con más formalidad, dando además un panorama de cómo pueden ser utilizadas como herramienta de clasificación. El capítulo 3 presenta en forma detallada la metodología elegida para la clasificación de los distintos patrones rítmicos, tanto de repique como de piano. Esto incluye el proceso de entrenamiento de las cadenas ocultas, y el proceso de clasificación a partir de las cadenas entrenadas. El capítulo 4 describe las pruebas realizadas y los resultados obtenidos, mientras que el 5 plantea algunas conclusiones y discusiones que surgen del desarrollo de la tesis, además de algunos caminos a recorrer que se abren de ahora en más. Finalmente, el apéndice A hace un breve resumen del código escrito para esta tesis, dando pautas para ejecutar los distintos ejemplos a los que se hará mención, y para reproducir los resultados reportados.

Capítulo 2

Cadenas de Markov y cadenas escondidas de Markov[1]

Consideremos un proceso estocástico en tiempo discreto $\{q_t\}$, para $t = 1, 2, 3, \dots$, donde q_t toma valores en un conjunto numerable S, que llamaremos espacio de estados. El proceso $\{q_t\}$ es una cadena de Markov si la probabilidad de transición entre estados depende únicamente del estado actual y del siguiente, es decir:

$$\mathbb{P}\left(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \dots\right) = \mathbb{P}\left(q_t = S_j | q_{t-1} = S_i\right).$$

En caso que la probabilidad de transición sea independiente del tiempo t, se dice que la cadena de Markov es homogénea, y escribimos:

$$a_{ij} = \mathbb{P}\left(q_t = S_j | q_{t-1} = S_i\right).$$

En este trabajo consideraremos cadenas homogéneas con espacio de estados finitos, es decir, el conjunto S es un conjunto con una cantidad finita N de elementos. Por lo tanto, podemos definir una matriz estocástica[2] A de dimensiones $N \times N$ cuyas entradas sean las probabilidades de transición a_{ij}, con $1 \leq i, j \leq N$. Si además definimos la distribución inicial de estados como un vector $\pi = \{\pi_i\}$, donde $\pi_i = \mathbb{P}(q_1 = S_i)$ con $1 \leq i \leq N$, entonces el proceso $\{q_t\}$ queda completamente determinado por la pareja (π, A).

Estas definiciones asumen que cada estado que puede tomar la cadena se corresponde a un evento físico (u observable). Las cadenas ocultas de Markov surgen como una extensión de las cadenas de Markov en ese sentido, dado que asumen que no es posible observar en qué estado se encuentra la cadena en cada instante, sino que la observación que se realiza es una función probabilística del estado.

En [56] se plantea un ejemplo simple de esta situación: supongamos que hay N urnas en una habitación, y cada urna contiene pelotas de M colores distintos (los colores no tienen por qué estar igualmente distribuidos en cada urna). Se

[1] El desarrollo de este capítulo está basado en el capítulo II de [56]

[2] Una matriz estocástica es aquella que cumple que todas sus entradas son no negativas y que la suma de los elementos de cualquiera de sus filas es igual a 1.

selecciona una urna para comenzar, y de la misma se extrae una pelota al azar. Se registra el color de la pelota y se devuelve la misma a la urna. Una nueva urna se elige al azar, y se repite el proceso de selección de pelotas. Así, se genera una secuencia de observaciones de colores. Para alguien que sólo conoce esa secuencia de observaciones, las urnas de las que fueron extraídas las pelotas no son conocidas (es decir, en este ejemplo la urna de la cual se extrajo cada pelota representa el estado no observable).

El ejemplo permite identificar de manera bastante clara cuáles son los elementos que caracterizan una HMM:

- N, la cantidad de estados en el modelo (en el ejemplo de las urnas, la cantidad de urnas en la habitación). De acuerdo a la notación que se venía utilizando, el conjunto de estados se denotará $S = \{S_1, S_2, \ldots, S_N\}$, y el estado en tiempo t será denotado q_t.

- M, la cantidad de símbolos de observación (en el ejemplo de las urnas, la cantidad de colores que pueden tener las pelotas). El conjunto de símbolos se notará $V = \{v_1, v_2, \ldots, v_M\}$.

- La distribución de probabilidad de transición de estados $A = \{a_{ij}\}$, donde:

$$a_{ij} = \mathbb{P}(q_{t+1} = S_j | q_t = S_i) \text{ con } 1 \leq i, j \leq N.$$

  En el ejemplo de las urnas, A sería la distribución de probabilidad con la que se selecciona la urna siguiente.

- La distribución de probabilidad de los símbolos de observación en el estado j, $B = \{b_j(k)\}$, donde

  $b_j(k) = \mathbb{P}(\text{observar } v_k \text{ en tiempo } t | q_t = S_j)$ donde $1 \leq j \leq N$ y $1 \leq k \leq M$.

  En el ejemplo de las urnas, B sería la distribución de colores en cada urna.

- La distribución inicial de estados $\pi = \{\pi_i\}$, donde:

$$\pi_i = \mathbb{P}(q_1 = S_i) \text{ con } 1 \leq i \leq N.$$

Por conveniencia, al igual que en [56], utilizaremos la notación $\lambda = (A, B, \pi)$. Así, dados N, M y λ, una HMM puede ser usada para generar secuencias de observaciones de la forma

$$O = O_1 O_2 \ldots O_T$$

donde cada observación O_t es alguno de los M símbolos de V, y T es el largo de la secuencia de observación. El proceso para generar una secuencia usando la HMM sería:

1) Elegir un estado inicial $q_1 = S_i$ de acuerdo a la distribución inicial de estados π.

2) Inicializar $t = 1$.

3) Elegir $O_t = v_k$ según la distribución de probabilidad de los símbolos de observación en el estado S_i, es decir, $b_i(k)$.

4) Realizar la transición a un nuevo estado $q_{t+1} = S_j$ de acuerdo a la distribución de probabilidad de transición de estados para el estado S_i, es decir, a_{ij}.

5) Actualizar $t = t + 1$ e ir a 3) si $t < T$; o finalizar el procedimiento si $t = T$.

Este procedimiento también puede pensarse como un modelo para explicar cómo una secuencia de observaciones dada fue generada por una cierta HMM.

2.1. Los tres problemas básicos de las HMMs

Si se quieren utilizar HMMs como una herramienta de modelado, surgen claramente tres problemas a enfrentar:

1. Dada una secuencia de observaciones $O = O_1O_2 \dots O_T$, y un modelo λ, ¿cómo se computa eficientemente $\mathbb{P}(O|\lambda)$, es decir, la probabilidad de la observación O dado el modelo?

2. Dada una secuencia de observaciones $O = O_1O_2 \dots O_T$, y un modelo λ, ¿cuál es la secuencia de estados correspondiente $Q = q_1q_2 \dots q_T$ que mejor explica las observaciones?[3]

3. ¿Cómo pueden ajustarse los parámetros $\lambda = (A, B, \pi)$ del modelo para maximizar $\mathbb{P}(O|\lambda)$?

El primer problema es uno de evaluación, es decir, dado un modelo y una secuencia de observaciones, cómo calcular la probabilidad de que esa secuencia haya sido producida por el modelo. Esto da una medida de qué tan bueno es el modelo a la hora de explicar una secuencia de observaciones dada. Si se está tratando de decidir entre distintos modelos, la solución al primer problema permite entonces elegir el modelo que mejor explica las observaciones.

El segundo problema intenta descubrir la parte oculta del modelo, esto es, hallar la secuencia "correcta" de estados. Si se piensa en el ejemplo de las urnas y las pelotas, dada una secuencia de observación de colores queda claro que no existe una única secuencia de urnas que explique esa secuencia de colores. De ahí el uso de comillas en la palabra"correcta": se debe definir un criterio de optimalidad para poder resolver este problema. La definición de dicho criterio se verá más adelante.

En el tercer problema se intenta optimizar los parámetros del modelo para describir de la mejor manera cómo fue generada la secuencia O. Es por lo tanto un problema de entrenamiento: dadas varias secuencias de observaciones (secuencias de entrenamiento), la solución a este problema permite ajustar iterativamente los parámetros del modelo para explicar de mejor manera cómo fueron generadas las observaciones.

[3]Más adelante se explicitará qué se quiere decir con "mejor explica".

En el presente trabajo se usarán varias HMMs para modelar cada uno de los patrones rítmicos introducidos en la sección anterior. Entonces, para cada uno, se deberá contar con una secuencia de entrenamiento que consista en varias realizaciones del patrón. La construcción de la HMM para cada patrón se hará resolviendo el problema 3. Una vez construidas las HMMs, la clasificación de una nueva secuencia se llevará a cabo usando la solución al problema 1: dada esa nueva secuencia, se calculará la probabilidad de que haya sido producida por cada uno de los modelos, y se clasificará la secuencia como perteneciente a la clase cuya HMM mejor explique las observaciones.

2.2. Solución a los tres problemas básicos de las HMMs

Se verá a continuación la solución a los tres problemas planteados en la sección anterior.

2.2.1. Solución al problema 1

Se quiere calcular la probabilidad de la secuencia de observación $O = O_1 O_2 \dots O_T$ dado el modelo λ, es decir, $\mathbb{P}(O|\lambda)$. La forma más directa es enumerar todas las posibles secuencias de estados de largo T. Si consideramos una secuencia de estados fija

$$Q = q_1 q_2 \dots q_T$$

entonces la probabilidad de observar la secuencia O dada la secuencia de estados Q es:

$$\mathbb{P}(O|Q,\lambda) = \prod_{t=1}^{T} \mathbb{P}(O_t|q_t,\lambda)$$

donde se asumió que las observaciones son independientes. Entonces:

$$\mathbb{P}(O|Q,\lambda) = b_{q_1}(O_1) b_{q_2}(O_2) \dots b_{q_T}(O_T).$$

Por otro lado, la probabilidad de la secuencia de estados Q dado el modelo es:

$$\mathbb{P}(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}.$$

Usando la regla de Bayes tenemos:

$$\mathbb{P}(O,Q|\lambda) = \mathbb{P}(O|Q,\lambda)\mathbb{P}(Q|\lambda).$$

La probabilidad de O se calcula sumando sobre todas las posibles secuencias de estados:

$$\begin{aligned} \mathbb{P}(O|\lambda) &= \sum_{\text{todo } Q} \mathbb{P}(O,Q|\lambda) = \sum_{\text{todo } Q} \mathbb{P}(O|Q,\lambda)\mathbb{P}(Q|\lambda) \\ \Rightarrow \mathbb{P}(O|\lambda) &= \sum_{q_1,q_2,\dots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots\dots a_{q_{T-1} q_T} b_{q_T}(O_T). \end{aligned}$$

Si bien esta solución es correcta, es exhaustiva computacionalmente, ya que requiere de aproximadamente $2TN^T$ operaciones. Esto es porque en cada $t = 1, 2, \ldots, T$ hay N posibles estados que pueden alcanzarse (por lo que hay N^T secuencias de estados posibles), y para cada uno de esos estados se deben realizar aproximadamente $2T$ operaciones. Por ejemplo, esto implica que para $N = 5$ y $T = 100$, se deben realizar $2 \times 100 \times 5^{100} \approx 10^{72}$ operaciones.

Sin embargo, existe un método más eficiente para el cálculo de esta probabilidad, conocido como *forward-backward procedure* (o procedimiento hacia adelante-hacia atrás) [7, 8]. Todo comienza con la definición de la variable hacia adelante

$$\alpha_t(i) = \mathbb{P}(O_1 O_2 \ldots O_t, q_t = S_i | \lambda) \quad \text{para } 1 \leq i \leq N.$$

Esta variable representa la probabilidad de la observación parcial $O_1 O_2 \ldots O_t$ (hasta tiempo t) y de estar en el estado S_i en tiempo t, dado el modelo λ. Inductivamente, $\alpha_t(i)$ puede calcularse de la siguiente manera:

1) Inicialización:

$$\alpha_1(i) = \pi_i b_i(O_1) \quad \text{para } 1 \leq i \leq N.$$

2) Inducción:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad \text{para } 1 \leq t \leq (T-1) \text{ y } 1 \leq j \leq N.$$

3) Finalización:

$$\mathbb{P}(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

El primer paso es la aplicación directa de la definición de la variable $\alpha_t(i)$ para $t = 1$. El paso de inducción se explica observando el esquema de la Figura 2.1. Allí se muestra cómo puede ser alcanzado el estado S_j en tiempo $t+1$ desde todos los estados S_i.

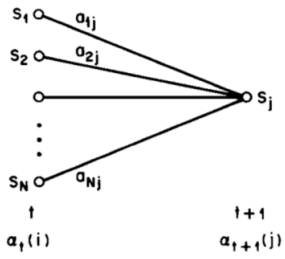


Figura 2.1: Formas de alcanzar el estado S_j en tiempo $t+1$ desde todos los estados S_i. Imagen extraída de [56].

Como $\alpha_t(i)$ es la probabilidad de observar el evento $O_1 O_2 \ldots O_t$ y de que el estado en tiempo t sea S_i, el producto $\alpha_t(i) a_{ij}$ representa la probabilidad de observar el evento $O_1 O_2 \ldots O_t$ y alcanzar el estado S_j en tiempo $t+1$ si el estado

en tiempo t es S_i. Al sumar sobre todos los estados posibles S_i, se tiene la probabilidad de estar en el estado S_j en tiempo $t+1$, habiendo observado $O_1O_2\dots O_t$. Para obtener $\alpha_{t+1}(j)$ resta incluir la observación O_{t+1} sabiendo que el estado en $t+1$ es S_j. De ahí la multiplicación de la suma por $b_j(O_{t+1})$, ya que esta es la probabilidad de observar O_{t+1} sabiendo que el estado es S_j. Finalmente, el paso 3 da la probabilidad $\mathbb{P}(O|\lambda)$ como la suma de las $\alpha_T(i)$, ya que, por definición:

$$\alpha_T(i) = \mathbb{P}(O_1O_2\dots O_T, q_T = S_i|\lambda).$$

y $\mathbb{P}(O|\lambda)$ puede verse como la suma del término de la derecha sobre todos los posibles estados S_i.

De este procedimiento se observa que la introducción de la variable α_t disminuye la cantidad de operaciones que es necesario realizar. En efecto, el cálculo de las $\alpha_t(i)$ requiere del orden de N^2T operaciones, frente a las $2TN^T$ que requería el cálculo directo. En el ejemplo con $N = 5$ y $T = 100$, se deben realizar aproximadamente 2500 operaciones, frente a las aproximadamente 10^{72} necesarias en la situación original.

Si bien la resolución del problema 1 se realiza utilizando solamente las variables hacia adelante, se introducirá ahora un procedimiento similar con variables hacia atrás, que será usado para la resolución del problema 3. Para cada estado, se define la variable hacia atrás:

$$\beta_t(i) = \mathbb{P}(O_{t+1}O_{t+2}\dots O_T|q_t = S_i, \lambda) \quad \text{para } 1 \leq i \leq N.$$

que representa la probabilidad de observar la secuencia $O_{t+1}O_{t+2}\dots O_T$ dado que en tiempo t la cadena se encuentra en el estado S_i y que se tiene el modelo λ. Nuevamente, los $\beta_t(i)$ pueden ser calculados inductivamente, siguiendo el siguiente procedimiento:

1) Inicialización:

$$\beta_T(i) = 1 \quad \text{para } 1 \leq i \leq N.$$

2) Inducción:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad \text{para } t = T-1, T-2, \dots, 1 \text{ y } 1 \leq i \leq N.$$

La justificación del paso de inducción se fundamenta de manera similar al del procedimiento hacia adelante. Si se observa la Figura 2.2, para haber estado en el estado S_i en tiempo t, y para cumplir con la secuencia de observación $O_{t+1}O_{t+2}\dots O_T$, se deben considerar todos los posibles estados S_j en tiempo $t+1$, transitar del estado i al estado j (de ahí el término a_{ij}), observar O_{t+1} en el estado j (de ahí el término $b_j(O_{t+1})$) y observar el resto de la secuencia $O_{t+2}\dots O_T$ (de ahí el término $\beta_{t+1}(j)$).

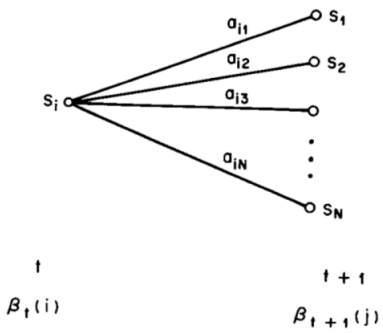


Figura 2.2: Formas de haber estado en el estado S_i en tiempo t desde todos los estados S_j en tiempo $t+1$. Imagen extraída de [56].

2.2.2. Solución al problema 2

A diferencia del problema 1, para el que se puede hallar una solución exacta, la solución al problema 2 depende del criterio de optimalidad que se considere, esto es, depende qué criterio se utilice para decir que la secuencia de estados hallada es la que "mejor explica" las observaciones. En la mayoría de las aplicaciones, el criterio que se utiliza es el de encontrar la secuencia de estados Q que maximiza $\mathbb{P}(Q|O,\lambda)$, o, equivalentemente, que maximiza $\mathbb{P}(Q,O|\lambda)$. A la secuencia hallada se la llama usualmente camino óptimo. La técnica para hallar este camino óptimo es conocida como algoritmo de Viterbi [25, 63].

Se llamará $Q = \{q_1, q_2, \ldots, q_T\}$ a la secuencia de estados y $O = \{O_1, O_2, \ldots, O_T\}$ a la de observaciones. El algoritmo de Viterbi comienza por definir la cantidad:

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} \mathbb{P}(q_1 q_2 \ldots, q_t = S_i, O_1 O_2 \ldots O_t|\lambda).$$

Es decir, $\delta_t(i)$ es la verosimilitud máxima entre todos los caminos que finalizan en el estado S_i en tiempo t, observando las primeras t observaciones de la secuencia de observaciones. Así, por inducción se tiene:

$$\delta_{t+1}(j) = \max_i [\delta_t(i)a_{ij}]\, b_j(O_{t+1}).$$

Como se quiere registrar la secuencia de estados que maximiza esta cantidad, se introduce una nueva variable $\psi_t(j)$. A partir de estas definiciones, el procedimiento para hallar el camino óptimo es:

1) Inicialización:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_1) \quad \text{para } 1 \le i \le N, \\ \psi_1(i) &= 0. \end{aligned}$$

2) Recursión:

$$\begin{aligned} \delta_t(j) &= \max_{1\le i\le N} [\delta_{t-1}(i)a_{ij}]\, b_j(O_t) \quad \text{para } 1 \le j \le N \text{ y } 2 \le t \le T, \\ \psi_t(j) &= \operatorname*{argmax}_{1\le i\le N} [\delta_{t-1}(i)a_{ij}] \quad \text{para } 1 \le j \le N \text{ y } 2 \le t \le T. \end{aligned}$$

3) Finalización:

$$\begin{aligned} P^* &= \max_{1\leq i\leq N} \left[\delta_T(i)\right], \\ q_T^* &= \operatorname*{argmax}_{1\leq i\leq N} \left[\delta_T(i)\right]. \end{aligned}$$

4) Backtracking:

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right) \quad \text{para } t = T-1, T-2, \ldots, 1.$$

Una vez finalizado este procedimiento, la secuencia de estados óptima es $Q^* = \{q_1^*, \ldots, q_T^*\}$.

En general, este algoritmo se implementa con una variante: como se está trabajando con probabilidades, la sucesión de multiplicaciones puede resultar en un *underflow*, es decir, que se obtenga un número demasiado pequeño como para ser representado por la máquina en la que se está realizando el cálculo. Esto puede evitarse si se trabaja con logaritmos, transformando así los productos en sumas. En ese caso, el procedimiento para el cálculo del camino óptimo es:

1) Inicialización:

$$\begin{aligned} \delta_1(i) &= \log\left(\pi_i\right) + \log\left(b_i(O_1)\right) \quad \text{para } 1 \leq i \leq N, \\ \psi_1(i) &= 0. \end{aligned}$$

2) Recursión:

$$\begin{aligned} \delta_t(j) &= \max_{1\leq i\leq N} \left[\log\left(\delta_{t-1}(i)\right) + \log\left(a_{ij}\right)\right] + \log\left(b_j(O_t)\right) \quad \text{para } 1 \leq j \leq N \text{ y } 2 \leq t \leq T, \\ \psi_t(j) &= \operatorname*{argmax}_{1\leq i\leq N} \left[\log\left(\delta_{t-1}(i)\right) + \log\left(a_{ij}\right)\right] \quad \text{para } 1 \leq j \leq N \text{ y } 2 \leq t \leq T. \end{aligned}$$

3) Finalización:

$$\begin{aligned} \log\left(P^*\right) &= \max_{1\leq i\leq N} \left[\delta_T(i)\right], \\ q_T^* &= \operatorname*{argmax}_{1\leq i\leq N} \left[\delta_T(i)\right]. \end{aligned}$$

4) Backtracking:

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right) \quad \text{para } t = T-1, T-2, \ldots, 1.$$

Esto también resulta adecuado porque en general las implementaciones del algoritmo devuelven, además del camino óptimo, su verosimilitud, es decir, la $\log\left(P^*\right)$. Además, los logaritmos de las matrices de transición de estados y de emisión de símbolos pueden precomputarse, de manera de ahorrar una operación en cada recursión.

2.2.3. Solución al problema 3

El tercer problema planteado es el de ajustar los parámetros $\lambda = (A, B, \pi)$ de manera de maximizar la probabilidad de una secuencia de observaciones dado el modelo. Si bien no existe una forma de resolver explícitamente este problema, es posible elegir λ de manera que $\mathbb{P}(O|\lambda)$ se maximiza localmente usando un procedimiento iterativo conocido como el algoritmo de Baum-Welch [6]. Para ello, definimos $\xi_t(i,j)$ como la probabilidad de estar en el estado S_i en tiempo t y en el estado S_j en tiempo $t+1$, dado el modelo y la secuencia de observaciones:

$$\xi_t(i,j) = \mathbb{P}(q_t = S_i, q_{t+1} = S_j | O, \lambda) = \frac{\mathbb{P}(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{\mathbb{P}(O|\lambda)}. \tag{2.1}$$

En la Figura 2.3 se muestra un diagrama de cómo tienen que ser las transiciones para que se cumplan esas condiciones. Así, se puede escribir $\xi_t(i,j)$ en función de

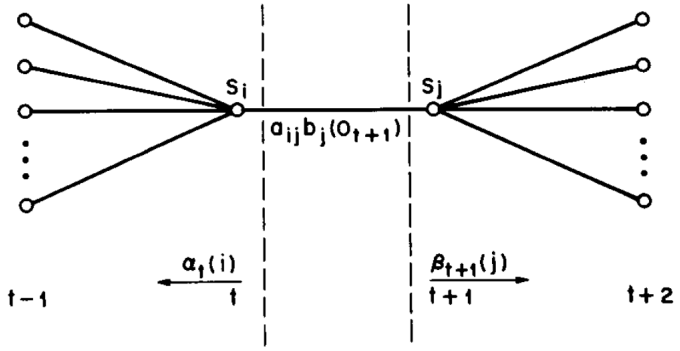


Figura 2.3: Formas de estar en el estado S_i en tiempo t y en S_j en tiempo $t+1$. Imagen extraída de [56].

las variables hacia adelante y hacia atrás:

$$\begin{aligned} \xi_t(i,j) &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\mathbb{P}(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}. \end{aligned}$$

Ahora se define una nueva cantidad $\gamma_t(i)$ como la probabilidad de estar en el estado S_i en tiempo t, dada una secuencia de observación y el modelo

$$\gamma_t(i) = \mathbb{P}(q_t = S_i | O, \lambda) = \frac{\mathbb{P}(q_t = S_i, O|\lambda)}{\mathbb{P}(O|\lambda)}. \tag{2.2}$$

Observemos que esta ecuación puede escribirse en función de las variables hacia adelante y hacia atrás:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^{N} \alpha_t(k)\beta_t(k)}.$$

Además, comparando 2.2 y 2.1 es claro que $\gamma_t(i)$ se relaciona con $\xi_t(i,j)$ mediante la ecuación:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j).$$

Si se suma $\gamma_t(i)$ sobre el índice de tiempo t se obtiene el número esperado (en el tiempo) de veces que el estado S_i es visitado. Equivalentemente, si se excluye el tiempo $t = T$ de la suma, esa cantidad puede verse como el número esperado de transiciones hechas desde el estado S_i. De igual forma, sumar $\xi_t(i,j)$ sobre t (con t desde 1 hasta $T-1$) indica la cantidad esperada de transiciones desde el estado S_i hacia el S_j. Entonces se puede obtener un método para reestimar los parámetros de una HMM, simplemente contando transiciones:

$$\bar{\pi}_i = \text{ cantidad de veces en estado } S_i \text{ en tiempo } 1 = \gamma_1(t),$$

$$\bar{a}_{ij} = \frac{\text{cantidad de transiciones desde } S_i \text{ a } S_j}{\text{cantidad de transiciones desde } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)},$$

$$\bar{b}_j(k) = \frac{\text{cantidad de veces en estado } S_j \text{ que se observa } v_k}{\text{cantidad de veces en estado } S_j} = \frac{\sum_{t=1 \text{ si } O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.$$

Si se define el modelo actual como $\lambda = (A, B, \pi)$, se usan esos parámetros para computar los términos de la derecha de las ecuaciones anteriores y se define el modelo reestimado como $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, donde $\bar{A}, \bar{B}$ y $\bar{\pi}$ son las cantidades dadas por los términos de la izquierda, en [8] se prueba que o bien el estimador inicial λ es un máximo en la función de verosimilitud, por lo que $\bar{\lambda} = \lambda$; o bien el modelo $\bar{\lambda}$ cumple que $\mathbb{P}(O|\bar{\lambda}) > \mathbb{P}(O|\lambda)$, por lo que se encuentra un nuevo modelo que explica mejor las observaciones.

Así, si se repite este procedimiento iterativamente se mejora la probabilidad de observar O en el modelo hasta que se obtiene un punto crítico. Vale la pena aclarar que este punto es un máximo local, por lo que el procedimiento no asegura que efectivamente se encuentre un máximo global de la función de verosimilitud.

El razonamiento realizado hasta este punto asume que las observaciones están caracterizadas por símbolos discretos elegidos de un alfabeto finito, pero existen ecuaciones análogas a las anteriores para el caso continuo. Para darlas, es necesario asumir que la densidad de probabilidad de las observaciones es una mezcla finita de la forma:

$$b_j(O) = \sum_{m=1}^{M} c_{jm} \Pi[O, \mu_{jm}, U_{jm}] \quad \text{para } 1 \leq j \leq N$$

donde O es el vector de observaciones, c_{jm} es el coeficiente para la m-ésima compo-

nente de la mezcla en el estado j,[4] y Π es cualquier densidad elíptica (por ejemplo, normal), con vector de media μ_{jm} y matriz de covarianza U_{jm} (para la m-ésima componente de la mezcla en el estado j).

Así, en [31, 32, 41] se muestra que las fórmulas de reestimación para estos parámetros son:

$$\bar{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(j,m)} \ , \ \bar{\mu}_{jm} = \frac{\sum_{t=1}^{T} [\gamma_t(j,m) \cdot O_t]}{\sum_{t=1}^{T} \gamma_t(j,m)}$$

$$\bar{U}_{jm} = \frac{\sum_{t=1}^{T} \left[\gamma_t(j,m) \cdot (O_t - \mu_{jm}) (O_t - \mu_{jm})^t\right]}{\sum_{t=1}^{T} \gamma_t(j,m)}$$

donde $\gamma_t(j,m)$ es la probabilidad de estar en el estado j en tiempo t, si la m-ésima componente de la mezcla es la responsable por haber emitido el símbolo O_t, es decir:

$$\gamma_t(j,m) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \frac{c_{jm}\Pi[O_t, \mu_{jm}, U_{jm}]}{\sum_{m=1}^{M} c_{jm}\Pi[O_t, \mu_{jm}, U_{jm}]}$$

Vale la pena observar que $\gamma_t(j,m)$ es la generalización de $\gamma_t(j)$ dada por 2.2, en el sentido de que ambas coinciden en el caso de tener una única densidad discreta. La fórmula de reestimación para a_{ij} en este caso es la misma que en el caso discreto, ya que la cantidad de estados sigue siendo discreta.

Las fórmulas de reestimación para este caso pueden interpretarse de la siguiente manera: para c_{jm} se calcula la proporción entre la cantidad esperada de veces que el sistema está en el estado j usando la m-ésima componente de la mezcla y la cantidad de veces que el sistema está en el estado j. Para μ_{jm}, el numerador de la fórmula de reestimación para c_{jm} es pesado por la obervación, dando así el valor esperado de la porción de la observación que corresponde a la m-ésima componente de la mezcla. Algo similar sucede con la reestimación de U_{jm}.

[4] Para que cada densidad b_j esté normalizada, los coeficientes de la mezcla deben verificar que $\sum_{m=1}^{M} c_{jm} = 1$ para todo $1 \leq j \leq N$.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Metodología

En este capítulo se explica de manera detallada la metodología utilizada para el entrenamiento de las cadenas ocultas que luego serán utilizadas para clasificar los patrones rítmicos. También se detalla cómo se realiza este proceso de clasificación.

3.1. Entrenamiento

Ya hemos establecido que para clasificar cada uno de los patrones rítmicos se entrenará una HMM; la Figura 3.1 muestra un diagrama de bloques del proceso de entrenamiento. Vale la pena aclarar que, independientemente de qué unidad de tiempo se elija reconocer (pulsos o compases, según lo discutido en las secciones 1.2 y 1.4) o del tambor que se trate (piano o repique), el procedimiento es esencialmente el mismo.

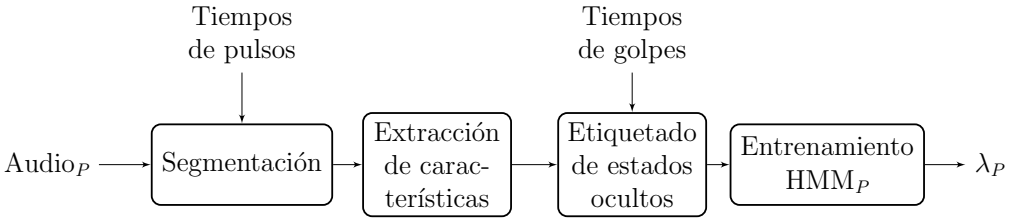


Figura 3.1: Diagrama de bloques del proceso de entrenamiento. Audio$_P$ representa el audio de entrenamiento asociado al patrón P y λ_P es el conjunto de parámetros de la HMM al finalizar el entrenamiento.

A continuación se realiza una descripción detallada de cada una de las etapas.

3.1.1. Segmentación

La etapa de segmentación, como su nombre lo indica, consiste en dividir la señal de audio en pulsos o compases, según cuál sea la unidad de tiempo a reconocer. Para audios sintéticos, los tiempos de comienzo de cada pulso (y por lo tanto de cada compás) están completamente determinados desde la generación de los datos,

por lo que el procedimiento es bastante simple: se divide el audio en partes según la duración de cada unidad.

En caso que también se usen audios reales, es necesario conocer la ubicación de los pulsos y eventualmente el inicio de cada compás. Actualmente existen algoritmos que realizan esta tarea automáticamente para grabaciones reales de ensambles de tambores [49]. Para este trabajo se cuenta con estos datos para todos los audios utilizados; la detección de tiempos musicales queda fuera del alcance de la tesis.

3.1.2. Extracción de características

Una vez segmentada la señal de audio, se realiza un procesamiento en tiempo corto bastante usual en audio para reconocimiento de timbre.

Las señales de audio de tambores de candombe presentan la dificultad de no tener información tonal bien definida; así, el reconocimiento de los distintos tipos de golpes debe centrarse en discriminarlos según el timbre de cada uno. Se entiende por timbre el atributo de un sonido que permite diferenciarlo respecto a otro con igual intensidad, altura (o frecuencia) y duración [2]. El timbre de un instrumento está determinado en gran medida por la distribución de energía en el espectro. Por esta razón, en la clasificación de instrumentos musicales es habitual el uso de características que describen el contenido espectral de un sonido [46, 57]. Los coeficientes cepstrales de frecuencias mel de la señal (MFCCs por sus siglas en inglés) [15] son tal vez las características más usadas para la descripción de timbre y son reconocidos por dar buenos resultados [11, 24, 42].

Así, se calculan los MFCCs dividiendo la señal de audio segmentada en tramas de 40 milisegundos, con un salto de 10 milisegundos (esto implica un 75 % de solapamiento entre tramas). Estos parámetros fueron elegidos empíricamente en base a las características de señales de audio de música de percusión. Se calcularon los 10 primeros MFFCs de cada trama. La elección de esta cantidad de coeficientes se hizo observando reconstrucciones de la señal original a partir de los coeficientes,[1] para distintas cantidades de coeficientes. En la Figura 3.2 se muestra el espectrograma de una señal del conjunto de entrenamiento junto al espectrograma de la reconstrucción de la señal usando 10 coeficientes. Allí se observa que con esa cantidad de coeficientes es posible reconstruir la señal de forma satisfactoria.

Al utilizar MFCCs como características para reconocimiento de voz es usual descartar el primer coeficiente, dado que es una medida global del nivel de energía de la señal y no es muy informativo en cuanto a su contenido espectral; así, si el sistema tiene que ser robusto a las variaciones de intensidad y funcionar para distintos hablantes grabados en condiciones variables, es razonable que sea descartado. Sin embargo, trabajos de detección de golpes en percusión como [55] no lo descartan, ya que, al dar una medida de la intensidad de la señal, puede servir

[1] Debe aclararse lo que se quiere decir con el término "reconstrucción". Los MFCCs no permiten reconstruir la señal original, ya que el proceso de cálculo implica transformaciones no invertibles. En realidad, lo que se hace es usar ruido blanco como excitación y observar la envolvente espectral extraída con los primeros coeficientes de MFCC. Por más detalle, ver [18].

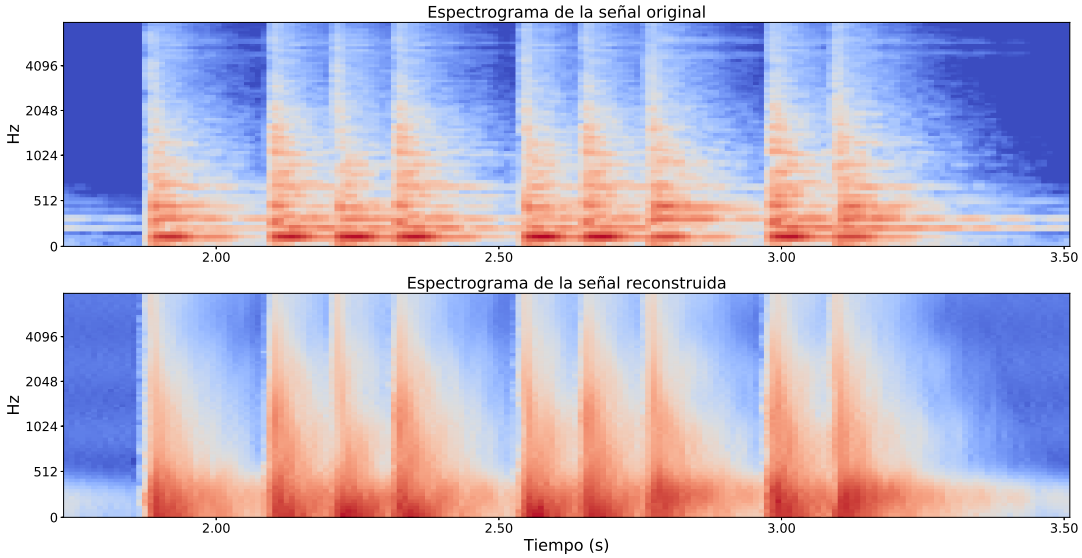


Figura 3.2: Espectrograma de una señal del conjunto de entrenamiento y espectrograma de la reconstrucción de la señal usando 10 MFCCs.

para indicar la presencia de un golpe. En este trabajo, se constató que incluir este coeficiente degrada el desempeño del sistema, por lo que se decidió descartarlo y usar los MFCCS 2 al 10.

Otra alternativa usual para detección de golpes en percusión es utilizar el flujo espectral (*spectral flux* en inglés) de la señal de audio, que es una medida de los cambios que se producen localmente en el espectro [47]. Si x_n es la n-ésima trama de audio, la transformada discreta de Fourier de esa trama es:

$$X(k,n) = \sum_{i=0}^{N-1} x_n(i) e^{-jki\frac{2\pi}{N}}$$

donde $k = 0, 1, \ldots, N-1$, siendo N el tamaño (en muestras) de la ventana utilizada para obtener las tramas.[2] A partir de la transformada discreta, el flujo espectral se calcula como:

$$\mathrm{SF}(n) = \frac{\sqrt{\sum_{k=0}^{N/2-1} \mathrm{HWR}(|X(k,n)| - |X(k,n-1)|)^2}}{N/2},$$

donde $\mathrm{HWR}(x)$ es la rectificación de media onda de la señal:

$$\mathrm{HWR}(x) = \frac{x + |x|}{2}.$$

[2]En este caso, como se usaron tramas de 40 ms, se tiene que $N = 40\,\mathrm{ms} * f_s$, siendo f_s la frecuencia de muestreo de la señal de audio. Por ejemplo, una f_s de 44,1 kHz corresponde a 1764 muestras.

La motivación para el rectificado de media onda es capturar las diferencias de magnitud del espectro sólo cuando son positivas, es decir, sólo cuando haya un aumento en la energía espectral. Dado que un golpe de percusión representa un gran aumento de energía en un período corto de tiempo, es esperable que se manifieste en el flujo espectral como un máximo local.

Así, se resolvió utilizar el uso del flujo espectral como característica, en lugar de utilizar el primer MFCC. De manera de lograr una característica independiente del nivel de la señal, el flujo espectral se normalizó utilizando un filtro de media móvil. Para ello, se utilizó una ventana móvil, y se calculó la norma 8 del flujo espectral en esa ventana. Cabe aclarar que los MFCCs también se normalizaron, de manera de que tengan media 0 y varianza 1 en todo el conjunto de entrenamiento. La normalización de los MFCCs se hizo de forma independiente para cada uno (se asume independencia en estas características).

Al final de este proceso, para cada trama de 40 ms se tiene un conjunto de 10 características: los 10 MFCCs si se mantiene el primero, o, si es descartado, los restantes 9 más el flujo espectral. En la Figura 3.3 se muestra un ejemplo de lo que resulta de esta etapa: para cada trama, se tiene una descripción del espectro (los MFCCs 2 a 10) más el flujo espectral para esa trama.

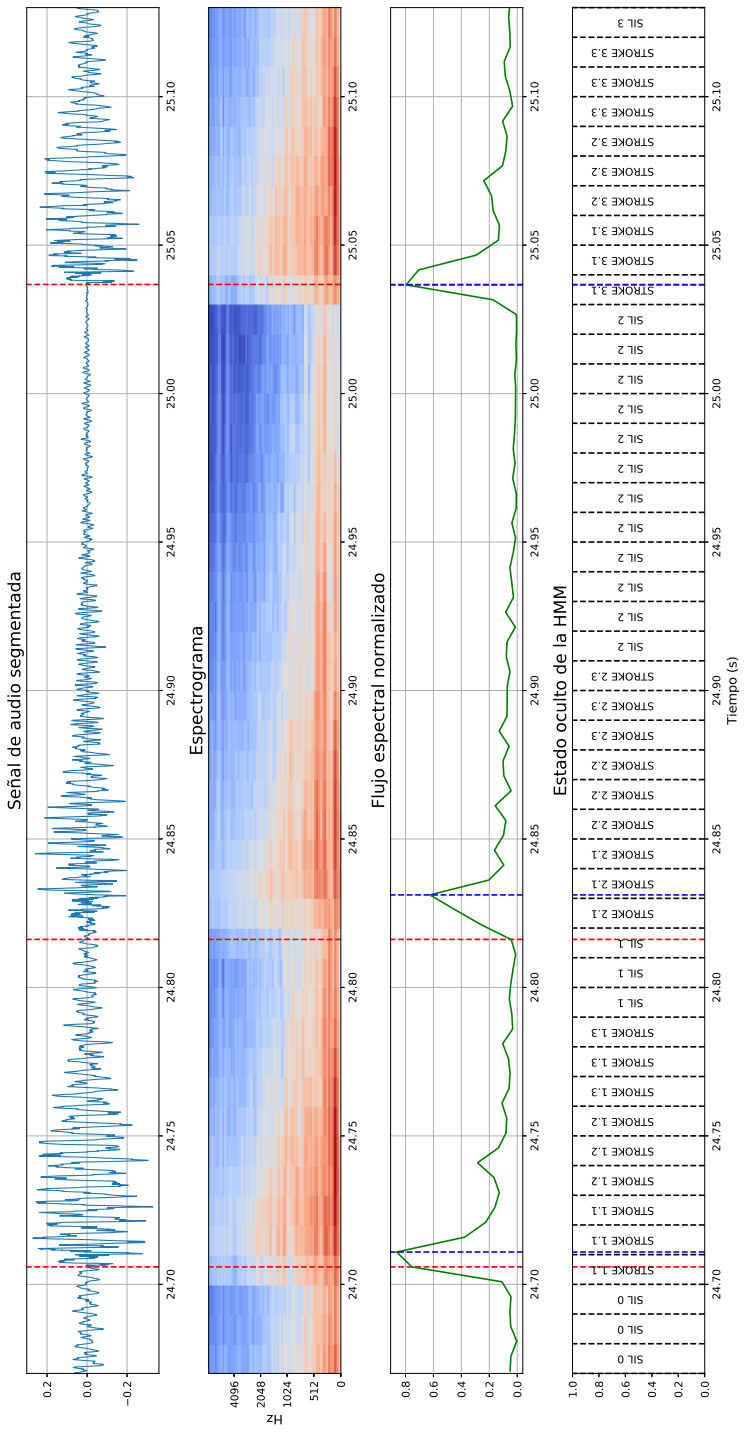


Figura 3.3: Señal de entrenamiento segmentada, espectro en escala mel, flujo espectral y división en tramas, con estados ocultos etiquetados. Las líneas punteadas rojas marcan la ubicación de los golpes, mientras que las azules marcan el máximo del *spectral flux* más cercano. El segmento de señal se corresponde a un patrón N_1.

3.1.3. Topología de las HMMs

Los estados ocultos de cada HMM se modelarán usando una cadena de izquierda a derecha, como es usual en reconocimiento de voz hablada [5, 28]. En la Figura 3.4 se muestra un diagrama de una de esas cadenas. El nombre proviene del hecho de que, a medida que el tiempo aumenta, el índice del estado oculto aumenta (o se mantiene). Así, el estado oculto se mueve de izquierda a derecha. Este tipo de topología intenta modelar señales cuyas propiedades varían a medida que el tiempo pasa, como sucede en el caso de la música, lo que la hace adecuada para la aplicación en cuestión.

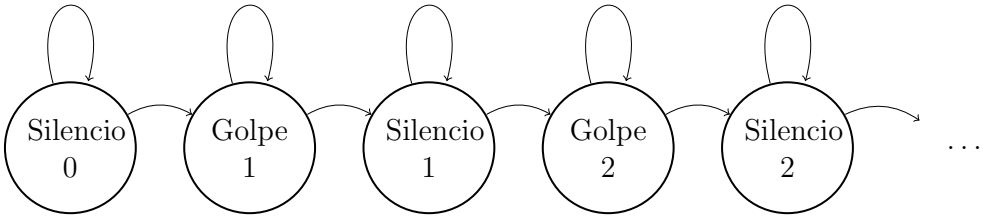


Figura 3.4: Diagrama de transiciones entre estados ocultos.

Matemáticamente, estas cadenas tienen una matriz de transición de estados cuyas entradas cumplen la propiedad:

$$a_{ij} = 0 \text{ si } i > j,$$

mientras que el vector de probabilidad de distribución inicial de estados tiene entradas:

$$\pi_i = \begin{cases} 1 & \text{si } i = 1 \\ 0 & \text{si } i \neq 1 \end{cases},$$

ya que la cadena debe comenzar en el primer estado. Además, debe finalizar en el último, por lo que la última fila de la matriz de transición debe cumplir:

$$a_{Ni} = \begin{cases} 1 & \text{si } i = N \\ 0 & \text{si } i \neq N \end{cases}.$$

En el caso del presente problema, la idea es que cada estado oculto represente la presencia de un golpe en la trama de audio o un silencio. Además, como se ve en la Figura 3.4, cada golpe estará asociado a un estado distinto, para modelar no solo el golpe sino la ubicación relativa del mismo respecto a los otros, aprovechando las ventajas ya mencionadas de las cadenas de izquierda a derecha. Así, las transiciones permitidas entre estados serán solamente de un estado al posterior, imponiendo sobre la matriz de transición la condición adicional:

$$a_{ij} = 0 \text{ si } j \neq i \text{ o } j \neq i + 1.$$

De manera de capturar la evolución temporal de las características durante la duración de un golpe, cada estado asociado a un golpe se modelará con tres sub-estados, como muestra la Figura 3.5. Se pretende que cada uno de estos sub-estados modele las distintas etapas de un golpe: ataque, sostenimiento y decaimiento.

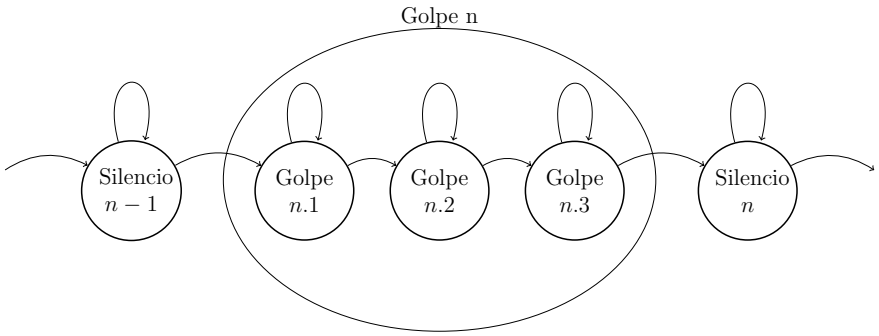


Figura 3.5: Diagrama de transiciones entre estados ocultos para un golpe.

Vale la pena aclarar que el término "sub-estado" se utiliza únicamente para explicar conceptualmente la topología de las cadenas usadas; no se corresponde con ningún objeto matemático asociado a las HMMs. A efectos prácticos, estos serán estados ocultos en su propio derecho.

3.1.4. Etiquetado de estados ocultos

Una vez definida la topología de las HMMs a utilizar, resta definir a qué estado oculto se corresponde cada observación del conjunto de entrenamiento. Esto es necesario para estimar los parámetros que describen la distribución de probabilidad de las observaciones, como fue explicado en el capítulo 2.

Como cada estado estará asociado a un golpe del tambor, para determinar en qué estado oculto se está en cada trama de audio es necesario conocer la ubicación de los golpes en la señal; se asume que esta información se conoce de antemano (lo que de hecho sucede para audios sintéticos). A partir de la ubicación de los golpes en el audio, la asignación o etiquetado de estados ocultos funciona como sigue: supongamos que, como se ve en la última fila de la Figura 3.3, el primer golpe está en la quinta trama.[3] Asumiendo que un golpe tiene 90 ms de duración, se toma una ventana de ese largo, con 10 ms antes del golpe y 80 ms después. Así, siguiendo con el ejemplo, las tramas 4 a 12 quedarían dentro de esta ventana.[4]

Luego, se divide la cantidad de tramas dentro de esta ventana entre la cantidad de sub-estados dentro de un golpe. En el ejemplo, hay nueve tramas a asignar y cada golpe está asociado a tres sub-estados, entonces estas nueve tramas se agruparán de a tres, asignando el primer grupo de tres al primer sub-estado, el segundo grupo al segundo sub-estado, y así sucesivamente.

Este procedimiento se repite para todos los golpes en el segmento de audio analizado. Finalmente, las tramas no asignadas se etiquetan como silencio, siguiendo el esquema de la Figura 3.5: las tramas vacías entre el golpe 1 y 2 se etiquetan como "Silencio 1", las que estén entre el segundo y tercer golpe se etiquetan como

[3]Para corregir posibles errores en el etiquetado de los golpes, cada golpe se ubica en el tiempo donde se da el máximo más cercano del *spectral flux*.

[4]Por la elección de parámetros para el cálculo de las características, las tramas se suceden cada 10 ms (40 ms de ventana de análisis con 1/4 de solapamiento).

"Silencio 2", etc. Aquellas que se encuentren antes del primer golpe se etiquetan "Silencio 0".

3.1.5. Entrenamiento de HMMs

La siguiente etapa del proceso consiste en entrenar las cadenas ocultas. Por "entrenar" se entiende el proceso de determinar los parámetros que describen la cadena, usando la solución al problema 3 vista en 2.2.3. Estos parámetros son: la distribución inicial de estados, la matriz de distribución de probabilidad de transición de estados y la distribución de probabilidad de las observaciones en cada estado.

Por la topología de las cadenas, la distribución inicial está fija: es 1 en el primer estado y ceros para el resto. Por las restricciones que ya fueron mencionadas en 3.1.3, la matriz de transición de estados solo tiene entradas no nulas en la diagonal y en la diagonal superior.[5] Así, sus entradas se obtienen simplemente contando la cantidad de veces que la cadena se mantiene en el mismo estado (para la diagonal), y la cantidad de transiciones entre un estado y el siguiente (para la diagonal superior).

Respecto a la distribución de probabilidad de las observaciones para cada estado, se asume que las mismas siguen una distribución normal. Dado que hay 10 observaciones por trama, esto implica que cada estado tiene asociado una densidad gaussiana en $\mathbb{R}^{10}$, de la que deben estimarse la media y la matriz de covarianza. Para que la cantidad de parámetros a estimar no sea excesiva, se asume que la matriz de covarianza es diagonal (es decir, se supone que las coordenadas del vector de observaciones son independientes). Esto implica que la matriz tiene únicamente 10 entradas no nulas, frente a las 100 que tendría si se utilizase una matriz de covarianza sin restricciones.

Para evaluar si la hipótesis de independencia es razonable, se realizó un gráfico Q-Q de las características en cada estado oculto. Este tipo de gráficos es utilizado para comparar visualmente dos distribuciones de probabilidad, y consiste en dibujar puntos cuyas coordenadas sean los cuantiles correspondientes a cada distribución.[6] Por ejemplo, supongamos que las distribuciones a comparar son gaussianas, una estándar (media 0, varianza 1) y la otra de media 1 (y manteniendo varianza 1). En la Tabla 3.1 se muestra la ubicación de los primeros tres cuantiles para cada una.

q	$\mathcal{N}(0,1)$	$\mathcal{N}(1,1)$
2 (mediana)	0	1
3 (terciles)	$(-0.431, 0.431)$	$(0.569, 1.431)$
4 (cuartiles)	$(-0.674, 0, 0.674)$	$(0.326, 1, 1.674)$

Tabla 3.1: Primeros cuantiles de una distribución normal estándar y una de media 1 y varianza 1.

[5] Se entiende por diagonal superior al conjunto de entradas a_{ii+1}.

[6] Su nombre proviene del inglés *Quantile-Quantile*.

Cada punto (x, y) del gráfico Q-Q se construye tomando como coordenadas x los valores de la segunda columna, y como coordenadas y los correspondientes de la tercera columna. Así, en el ejemplo el gráfico tendrá puntos en las coordenadas: $(0, 1)$, $(-0.431, 0.569)$, $(0.431, 1.431)$, $(-0.674, 0.326)$ y $(0.674, 1.674)$.

Este tipo de gráficos también puede utilizarse como medida de qué tanto se ajusta un conjunto de datos a una determinada distribución. Simplemente en un eje se usan los cuantiles teóricos de la distribución, mientras que en el otro se usan los estimados a partir de los datos. Si el resultado de esa gráfica es aproximadamente una línea recta, quiere decir que las distribuciones son similares (si fuesen exactamente iguales, todos los puntos caerían en la recta $y = x$).

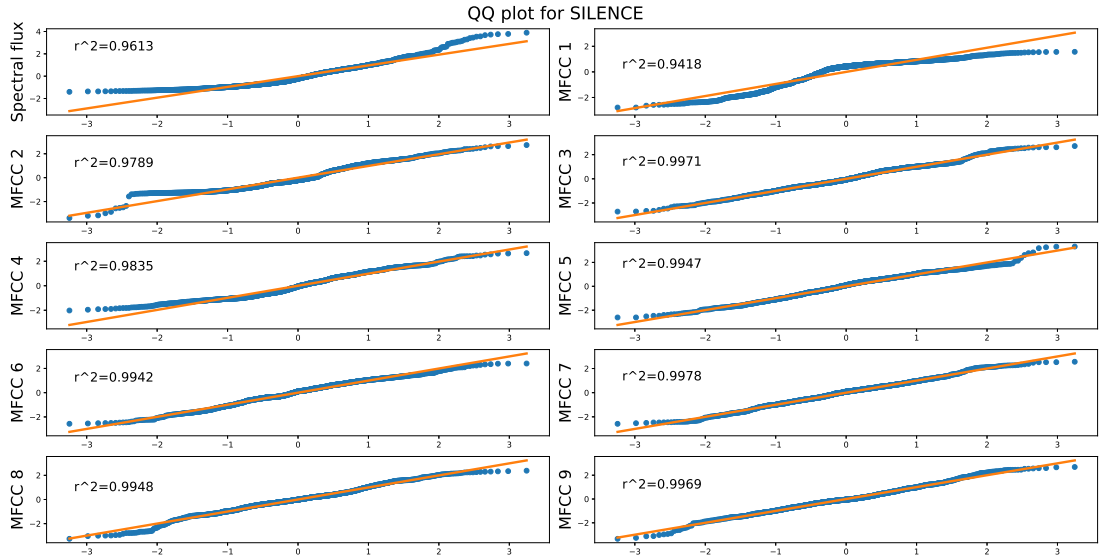
En la Figura 3.6 se muestran los gráficos para este caso en particular. Allí se pueden observar los gráficos obtenidos para las cuatro clases de estados ocultos (silencio y los tres sub-estados asociados a un golpe).

Los gráficos se construyen calculando las características con todos los datos disponibles (es decir, todos los audios etiquetados con los que se cuenta). Como se asume que la matriz de covarianza de la distribución en cada estado oculto es diagonal, el gráfico muestra la comparación de los cuantiles de cada una de la características frente a los de una normal (0,1).[7] En la Figura 3.6 también se ve el resultado de la regresión lineal de los puntos del QQ-plot, y el cuadrado del coeficiente de correlación de esa regresión. Como ya fue mencionado, si las distribuciones comparadas son similares, el resultado debería ser cercano a una recta (en el caso de dos distribuciones teóricas, es fácil probar que si una es una transformación afín de la otra, entonces los cuantiles también se relacionan mediante una transformación afín). Esto explica el ajuste lineal realizado.

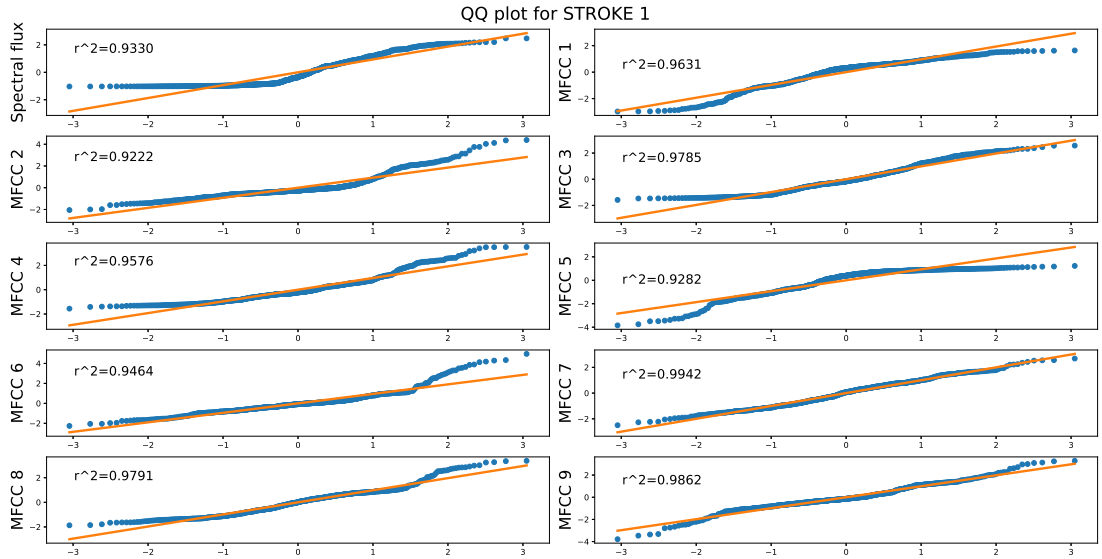
Analizando los gráficos, se ve que en general la aproximación es razonable. La mayor discrepancia entre la distribución esperada y la real se encuentra en el flujo espectral. Esto es especialmente cierto para los estados "Silencio" y "Stroke 2" (primer gráfico de 3.6a y 3.6c). Esto probablemente se deba a que en algunos estados considerados "Silencio", especialmente aquellos inmediatamente anteriores al comienzo de un golpe, el espectro ya haya empezado a modificarse debido a la presencia de ese golpe. Eso implica que será considerada como "Silencio" al menos una trama de audio en el que en realidad se está desarrollando el ataque de un golpe.

Más allá de estas consideraciones, los gráficos de la Figura 3.6 muestran que no es descabellado asumir distribuciones normales con matrices de covarianza diagonales. En todo caso, siempre cabe la posibilidad de utilizar otras distribuciones, como por ejemplo una mezcla de gaussianas. Esto conllevaría un aumento de los parámetros a estimar, lo que es un problema dada la escasez de datos con los que se cuenta. Además, es inconveniente por motivos prácticos: el toolbox utilizado, si bien cuenta con la opción de utilizar mezclas (aunque solamente de gaussianas), tiene problemas de implementación en la estimación de parámetros. Esto causa que los métodos iterativos descriptos en el Capítulo 2 no converjan, impidiendo

[7] Las características son normalizadas para que la comparación tenga sentido.



(a) Silencio

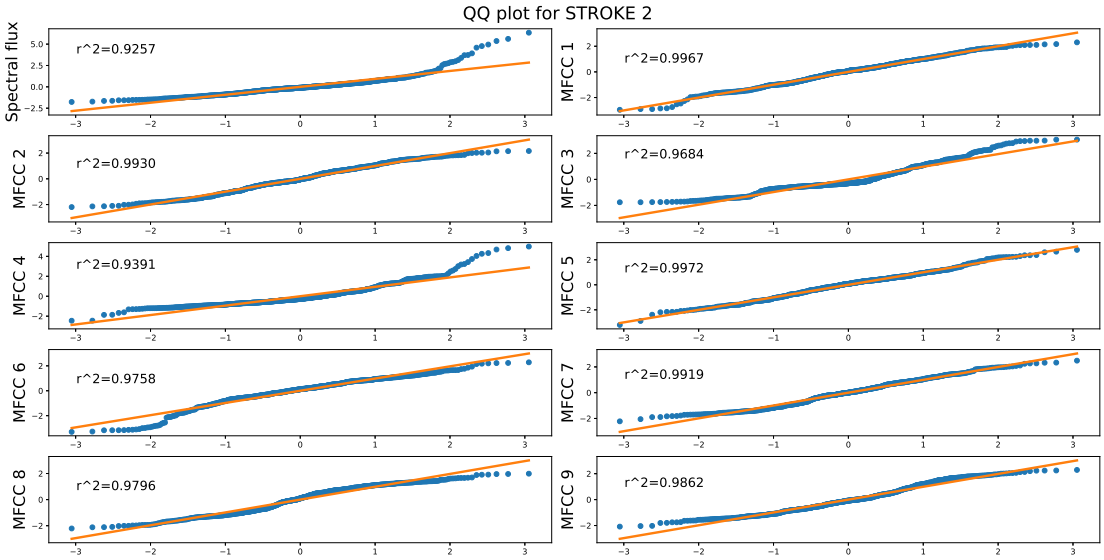


(b) Golpe (sub-estado 1)

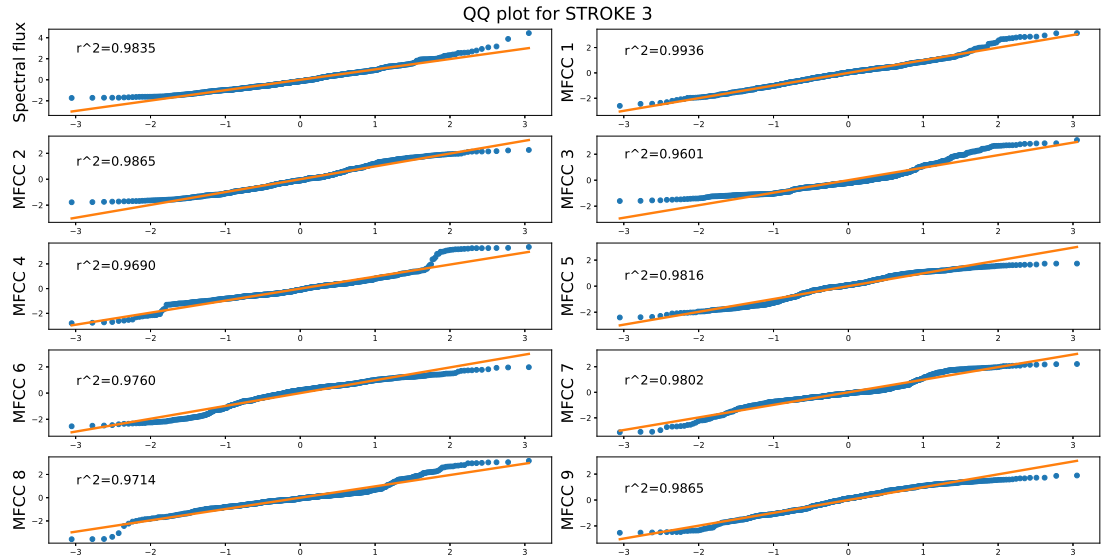
Figura 3.6: QQ-plot para las características de los distintos estados ocultos. Los cuantiles teóricos se ubican en el eje x.

así el ajuste correcto a los datos, y perturbando el proceso de entrenamiento.

Una vez finalizada la etapa de entrenamiento, se tienen completamente determinados los parámetros que constituyen cada HMM, por lo que se puede realizar la clasificación de una nueva señal de audio. En la sección que sigue, se explica el proceso de clasificación para cada problema abordado en la tesis.



(c) Golpe (sub-estado 2)



(d) Golpe (sub-estado 3)

Figura 3.6: QQ-plot para las características de los distintos estados ocultos. Los cuantiles teóricos se ubican en el eje x.

3.2. Clasificación

3.2.1. Variantes de repique

En la Figura 3.7 se muestra un diagrama de bloques del proceso de clasificación de patrones de repique, usando las cadenas previamente entrenadas.

Dada una señal de audio a clasificar, las primeras dos etapas de procesamien-

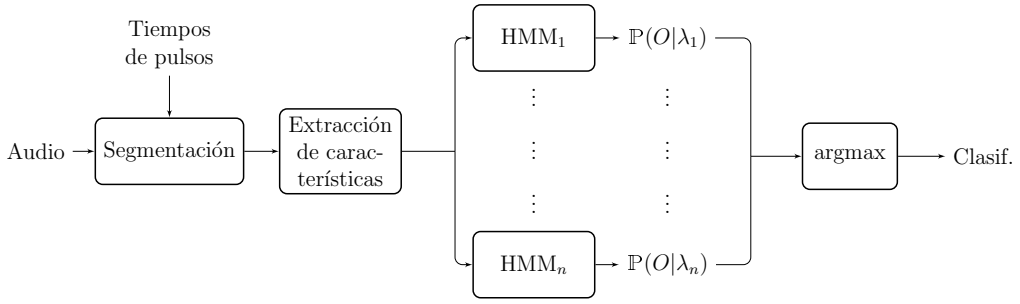


Figura 3.7: Diagrama de bloques del proceso de clasificación de patrones de repique.

to coinciden con las que se realizan en el proceso de entrenamiento. Esto es: se segmenta la señal en la unidad de tiempo elegida y se extraen las características para la señal segmentada. Luego, se calcula la probabilidad de esa secuencia de observaciones según cada HMM. Esto se hace usando la solución al problema 1 visto en la Sección 2.2.1. Finalmente, se clasifica el compás o el pulso (según la unidad de tiempo elegida) como perteneciente a la clase cuya HMM da la mayor probabilidad de haber observado esa secuencia de símbolos.

3.2.2. Base de piano

En el caso del piano, el objetivo es identificar si un compás consiste en la ejecución de la base de piano (o alguna variante de esta), o si en cambio es un compás improvisado. Así, el problema es uno de clasificación binaria: un compás pertenece (o no) a la clase "Base". Esto es una diferencia clave con lo que se hace para el repique, donde se asume que un compás pertenece a una de las clases conocidas (o se asignará a una de ellas). Por lo tanto, en este caso se tiene una única cadena, entrenada para reconocer al patrón base. Así, el proceso de clasificación será un poco diferente al que se realiza para el repique. En la Figura 3.8 se muestra un diagrama de bloques para este caso.

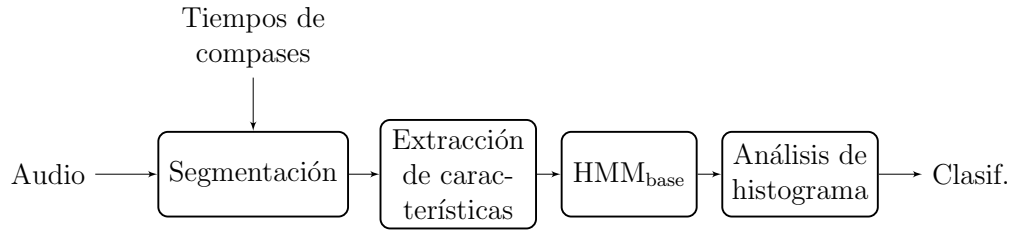


Figura 3.8: Diagrama de bloques del proceso de clasificación de la base del piano.

Allí se ve que las primeras tres etapas del proceso son análogas al caso del repique, con la salvedad de que aquí no hay cadenas en paralelo. Estas etapas son: segmentación del audio en compases, extracción de características y cálculo de la probabilidad de la secuencia de observaciones dado el modelo entrenado.

Luego, se realiza un análisis del histograma de las verosimilitudes (en realidad,

de las log-verosimilitudes) obtenidas. Este análisis consiste en calcular un histograma acumulativo normalizado de las log-verosimilitudes para todos los compases que se quieran clasificar. Se clasificarán como "Base" aquellos compases para los que el histograma normalizado supere cierto umbral previamente definido. Si por ejemplo este umbral es de 0.19, se clasificarán como "Base" los compases en los que el histograma normalizado supere el valor 0.19. El resto de los compases se clasificarán como "variaciones" del patrón. En la Figura 3.9 se muestra gráficamente esta situación para un audio en particular.

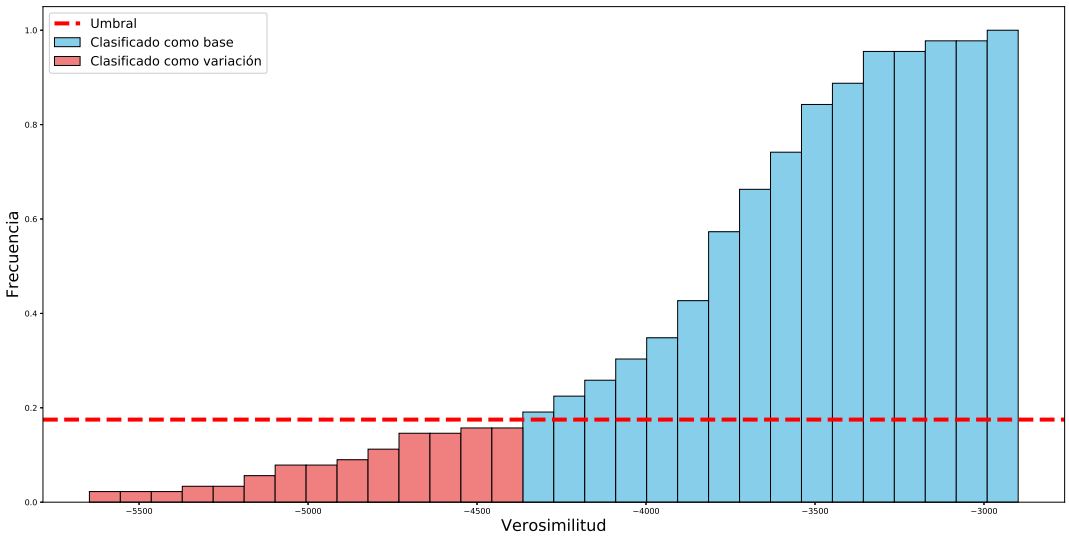


Figura 3.9: Histograma acumulativo normalizado de las verosimilitudes y clasificación de cada compás, para un audio particular.

La determinación del umbral se realiza de la siguiente manera: se llevan a cabo las primeras tres etapas del proceso de clasificación sobre todos los audios utilizados para el entrenamiento.[8] Una vez que se tiene la log-verosimilitud para cada compás de los audios que se usaron para entrenar, se calcula un histograma acumulativo normalizado de estas cantidades para cada audio. Luego, se evalúa el impacto que tienen distintos niveles de umbralización sobre este histograma, y se elige el nivel que sea máximo para todo el conjunto de entrenamiento. Dado que este es un problema de clasificación binario (un compás es "Base" o no lo es), la evaluación de impacto consiste en calcular el $f-$measure asociado a los distintos umbrales. Así, se elige el umbral como el que haya dado una mejor $f-$measure sobre todos los audios del conjunto de entrenamiento.

Se recuerda que, en un esquema de clasificación binario, el $f-$measure se define como la media armónica del *precision* p y el *recall* r. Estas cantidades se definen

[8]Cabe aclarar que aquí se utilizan los audios de entrenamientos en su totalidad, a diferencia del entrenamiento, donde se utilizaban solamente los compases etiquetados como "Base", ya que la cadena se entrena para reconocer ese patrón.

como:

$$p = \frac{tp}{tp + fp} \ , \ r = \frac{tp}{tp + fn}$$

donde tp (abreviación de *true positives*) es la cantidad de instancias correctamente clasificadas como "Base", fp (abreviación de *false positives*) es la cantidad de instancias incorrectamente clasificadas como "Base", y fn (abreviación de *false negatives*) es la cantidad de instancias incorrectamente clasificadas como "variaciones" (es decir, pertenecen a la clase "Base" pero no se clasifican como tal). Por lo tanto, el *precision* mide la proporción entre la cantidad de compases bien clasificados como "Base", respecto al total de compases clasificados así. En ese sentido, es una medida de cuántos compases clasificados como "Base" efectivamente lo son, pero no dice nada respecto a la cantidad de compases de "Base" que son mal clasificados. Esta información es complementada por el *recall*, que es la proporción entre las instancias correctamente clasificadas como "Base" y el total de instancias de la clase "Base". Por lo tanto, ambas medidas se complementan para dar una noción de qué tan buena es la clasificación: un *precision* de 1 indica que todas los compases clasificados como "Base" efectivamente lo son, mientras que un *recall* de 1 dice que todos los compases de "Base" son bien clasificados (pero no dice nada sobre cuántos que no son "Base" se clasifican como tal). El f-measure, al ser la media armónica de estas dos cantidades, combina lo que ambas dicen sobre la clasificación para dar una medida global del desempeño del sistema, y se calcula:

$$f = \frac{2}{\frac{1}{p} + \frac{1}{r}} = 2\frac{pr}{p + r}.$$

Capítulo 4

Resultados experimentales

En este capítulo se explican las pruebas realizadas para evaluar el desempeño del sistema de clasificación propuesto, tanto para la clasificación de variantes de repique como para el reconocimiento de la base de piano. Tanto el proceso de entrenamiento como el de clasificación fue escrito en Python, usando las bibliotecas `librosa` para el procesamiento de audio y `hmmlearn` para todo lo relacionado con cadenas ocultas. En el apéndice A se explica en detalle todo lo referente al código escrito, junto con instrucciones para reproducir los resultados que aquí se exponen.

4.1. Clasificación de variantes de repique

Al intentar realizar pruebas en este contexto aparece el problema de la falta de datos adecuados, es decir, la falta de grabaciones de candombe que cuenten con las condiciones adecuadas. Idealmente, las grabaciones deberían ser sin ruido de ambiente (las grabaciones de candombe más comunes se realizan en la calle, lo que implica un elevado nivel de ruido), sin ruido cruzado (grabaciones con un tambor por canal, que en los contextos callejeros resulta casi imposible) y etiquetadas (esto implica ubicación temporal de pulsos y compases y asignación de cada uno a las clases que se quiere reconocer). Si las primeras condiciones resultan difíciles de cumplir, el etiquetado lo es aún más, por dos motivos: primero, el problema de clasificación de toque de candombe es muy específico como para que haya un trabajo amplio de etiquetado en el tema; segundo, porque el toque de repique y piano es de una riqueza infinitamente mayor a las pocas variantes aquí consideradas, por lo que en una grabación real comúnmente solo se encuentran unos pocos compases que se ajustan a las clases analizadas. Esto último es un problema sobre todo para el entrenamiento, ya que son necesarias varias instancias de los patrones analizados para ajustar los parámetros de las cadenas ocultas, y en una grabación real los patrones "aprovechables" son realmente escasos. Este problema se agudiza en el caso del repique, donde los patrones a clasificar (y por lo tanto, las cadenas a entrenar) son varios, por lo que las pocas instancias "aprovechables" se dividen entre los distintos patrones, disminuyendo aún más el conjunto de datos con el que se puede entrenar cada cadena.

Fue por ese motivo que surgió la idea de utilizar audios sintéticos. La ventaja es que pueden generarse tantas interpretaciones como sean necesarias, donde aparezcan únicamente los patrones de interés (o pequeñas variaciones de ellos). La desventaja es que estas interpretaciones son muy "de laboratorio", en el sentido que no cuentan con las pequeñas desviaciones temporales que le dan el *swing* característico al candombe,[1] además de que las condiciones acústicas de esos audios difícilmente se parezcan a las de una grabación real, aún si esta se realiza en las mejores condiciones. En las pruebas que se describen a continuación se tuvo en cuenta este problema, analizando cómo es el desempeño del sistema si se usan audios sintéticos para el entrenamiento y se intentan clasificar audios reales. Como se verá más adelante, la clasificación en ese caso no es buena.

Los audios sintéticos fueron generados usando Lilypond [3] (para la generación de partituras a partir de las que se sintetizaron los audios) y Csound [1] (para la síntesis). Para generarlos, primero se escribe una partitura en Lilypond. Esa partitura es parseada por un programa en Python, que genera una orquesta de Csound. Los golpes usados por la orquesta en la síntesis son seleccionados al azar de un grupo de muestras previamente grabadas por un músico profesional, y el programa se encarga de intercalar esos golpes pregrabados según lo que indique la partitura. Por lo tanto, el sonido de la interpretación no es sintético (puesto que el sonido de los golpes proviene de grabaciones reales), pero la interpretación sí lo es (ya que se genera a partir de una partitura que no proviene de una interpretación real).

El proceso incluye la generación de un archivo de audio en formato `.wav`, la creación de un archivo de texto indicando los tiempos de inicio de cada compás (en ese archivo también se indican los tiempos de cada *tactus*) y de otro archivo indicando la ubicación temporal de los golpes. Cabe destacar que los audios generados cuentan con los patrones tal cual los define Jure en [34], esto es, los patrones aparecen sin las modificaciones o adornos discutidos en la sección 1.2. Esta decisión se tomó pensando en evaluar si el sistema de clasificación diseñado es capaz de absorber esas modificaciones.

El esquema de pruebas planteado fue de complejidad incremental. Primero, se evaluó el funcionamiento del sistema al clasificar un archivo de audio sintético en el que solo aparecen los patrones originales, sin adornos. Denominaremos a este archivo de audio ASR1, por "archivo de audio sintético de repique número 1". Luego, se evaluó utilizando un audio sintético en el que no solo aparecen los patrones originales, sino que también hay algunos adornados. Usando el mismo criterio que antes, denominaremos a este archivo ASR2. El audio ASR1 es una síntesis hecha a partir de una interpretación real, grabada en un toque callejero y pautada por Jure en [34].[2] La síntesis se hizo eliminando los adornos de la pieza. Para la segunda prueba no se usó el audio de Jure, sino que se sintetizó uno nuevo. Si bien la interpretación pautada por Jure tiene algunos adornos, muchas de las

[1] Por más información sobre este punto, un artículo que trata sobre las pequeñas desviaciones temporales en candombe es [37].

[2] El intérprete de esa grabación es José Pedro "Perico" Gularte, reconocido tocador de repique de Ansina.

posibles modificaciones no aparecen, por lo que se optó por generar un nuevo audio que contara con más variantes.

Finalmente, se realizó una prueba clasificando otra grabación real también pautada por Jure. Esta grabación no es la misma que se utilizó para sintetizar el audio ASR1. La usada en este caso cuenta con mejores condiciones acústicas que la pautada por Jure en [34], ya que fue grabada en estudio, con un canal por tambor, minimizando así el ruido cruzado.

4.1.1. Entrenamiento y clasificación con audio sintético

Como ya fue mencionado, la primera etapa de evaluación fue realizada enteramente con audio sintético. Para el entrenamiento, se sintetizaron cuatro archivos de audios de 17 compases cada uno. La síntesis se hizo solo usando los patrones originales, sin los adornos discutidos en 1.2. Esos audios fueron los utilizados para entrenar las distintas HMMs.

Para la clasificación se usaron los audios ASR1 y ASR2. En la figura 4.1 se muestra la partitura del ASR1; como puede apreciarse, no aparece allí ningún patrón adornado.

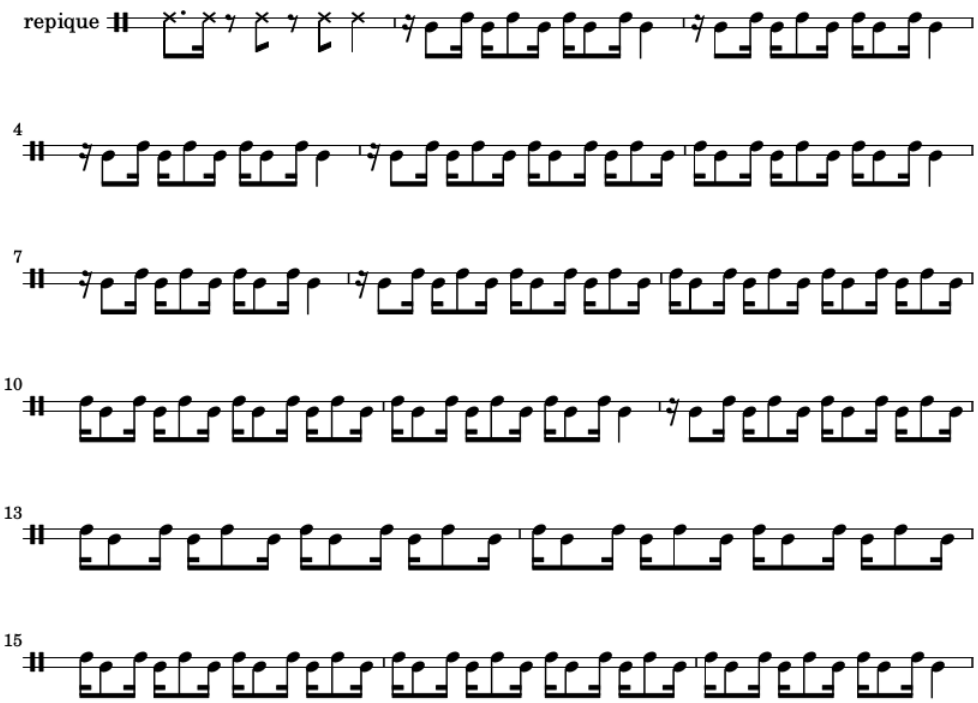


Figura 4.1: Partitura del audio ASR1, usado para la primera prueba de clasificación.

El segundo audio utilizado, denominado ASR2, no se corresponde con una interpretación real. No se utilizó una síntesis de la interpretación pautada por Jure, ya que este no cuenta con suficientes variantes como para evaluar el alcance del enfoque elegido. Fue sintetizado tratando de incluir la mayor cantidad de adornos y

variantes de los patrones de entrenamiento; su partitura se muestra en la figura 4.2. Como allí se ve, además de las variantes aparecen algunos patrones originales. La partitura se escribió de manera que la interpretación fuese musicalmente coherente (por ejemplo, si en un compás comenzó un repicado, en el siguiente se continúa o se resuelve, pero no comienza un repicado nuevo).

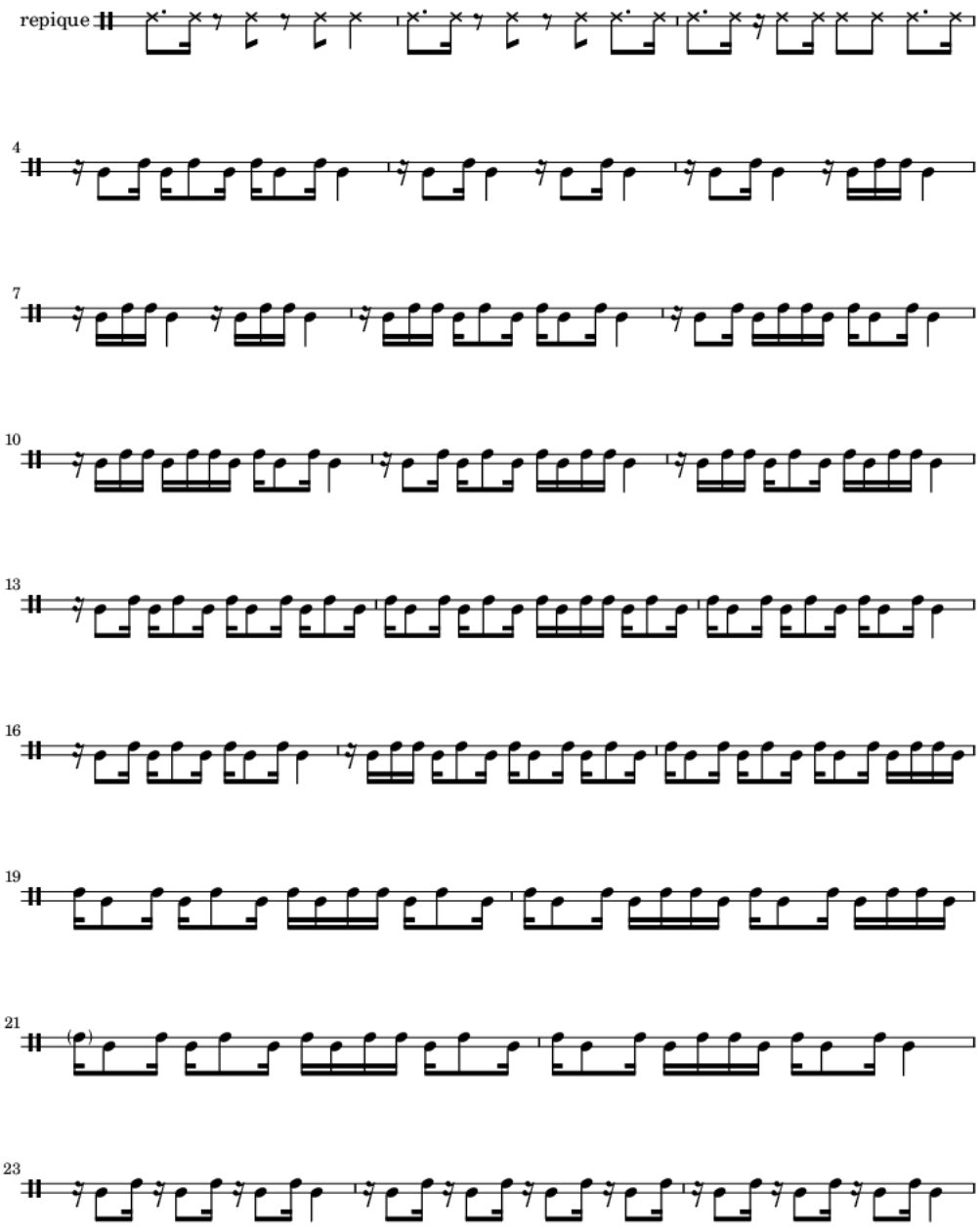


Figura 4.2: Partitura del audio ASR2, usado para la segunda prueba de clasificación.

A continuación se presentan los resultados para la clasificación de ambos au-

dios, según la unidad rítmica que se intente reconocer (pulsos o compases).

Clasificación de compases

En el caso de clasificación de compases, se tienen cinco patrones a reconocer: madera (M), axioma (A), comienzo (C), resolución (R) y liso (L), según lo discutido en la sección 1.2. Cabe aclarar que en esta clasificación se mantienen los compases de madera de la pieza, mientras que en la clasificación de pulsos esos compases no se utilizan. Esta diferencia se debe a que, a nivel de pulsos, la madera presenta muchas variantes en la ubicación de los golpes (véanse, por ejemplo, los primeros tres compases de 4.2), por lo que hace más sentido tratarla a nivel de compases. En la práctica, esto no presenta un problema, ya que los compases de madera pueden tratarse separadamente de manera relativamente simple (como en [36, 59]).

Hecha esta aclaración, los resultados obtenidos para compases en el caso de reconocimiento de audios sintéticos se presentan a continuación. Para la primera prueba (la de los patrones sin adornar), se obtuvo un 100 % de acierto en la clasificación, como se muestra en la tabla 4.1. Allí se muestra, para cada compás del archivo ASR1, a qué clase pertenece (segunda columna) y a qué clase fue asignado en la clasificación (tercera columna). Recordemos que la asignación se realiza a la clase cuya HMM haya dado la log-verosimilitud más alta; en la cuarta columna se muestra qué clase fue la que obtuvo la segunda log-verosimilitud más alta. Esto se reporta de manera de tener una idea de qué tan discriminante es el sistema: si el compás pertenece claramente a alguna de las clases, la log-verosimilitud más alta debería ser mucho mayor que la asociada a los otros patrones.[3]

Se reporta además una métrica que pretende contribuir en esa discusión. En la quinta columna se muestra la diferencia relativa entre el la log-verosimilitud máxima y la correspondiente al siguiente máximo. Si x_{max} es la log-verosimilitud máxima y $x_{\text{next-max}}$ es la segunda log-verosimilitud más alta, la diferencia relativa se calcula de la siguiente manera:

$$d_{\text{rel}} = \frac{|x_{\text{next-max}} - x_{\text{max}}|}{|x_{\text{next-max}}|} = \left|1 - \frac{x_{\text{max}}}{x_{\text{next-max}}}\right|$$

Dado que $x_{\text{next-max}} \leq x_{\text{max}} \leq 0$, el cociente $\frac{x_{\text{max}}}{x_{\text{next-max}}}$ está entre 0 y 1 y por lo tanto la diferencia relativa está entre 0 y 1. Además, cuanto más cercanas sean ambas log-verosimilitudes, más cercano a 1 es ese cociente, haciendo que la diferencia relativa sea cercana a 0 (y análogamente para el caso en el que las log-verosimilitudes sean muy diferentes).

Así, esta medida representa de alguna manera la "confiabilidad" de la clasificación: cuanto más cercana a 1 sea, más poder de discriminación tiene el sistema al clasificar ese compás, y más confiable es la clasificación. Como se aprecia en la

[3]Cabe recordar que las log-verosimilitudes siempre son menores o iguales que cero, por lo que se habla de mayor en el sentido usual: la mayor no es la más grande en valor absoluto, si no la más cercana a 0.

Compás	*Ground truth*	*Clasificación (score)*	*Siguiente máximo (score)*	*Dif. relativa*
1	M	M (-1884.94)	C (-16523.42)	0.886
2	A	A (-628.81)	C (-1620.65)	0.612
3	A	A (-250.32)	C (-1695.49)	0.852
4	A	A (-355.15)	C (-1675.48)	0.788
5	C	C (-378.76)	A (-911.38)	0.584
6	R	R (-364.57)	L (-1346.79)	0.729
7	A	A (-391.08)	C (-1627.84)	0.760
8	C	C (-303.42)	A (-689.47)	0.560
9	L	L (-189.62)	R (-956.00)	0.802
10	L	L (-204.44)	C (-820.53)	0.751
11	R	R (-329.40)	L (-1547.24)	0.787
12	C	C (-372.23)	A (-904.07)	0.588
13	L	L (-243.26)	R (-801.03)	0.696
14	L	L (-240.29)	R (-839.05)	0.714
15	L	L (-383.64)	R (-1106.97)	0.653
16	L	L (-186.92)	R (-843.11)	0.778
17	R	R (-336.13)	L (-1443.29)	0.767

Tabla 4.1: Resultados de la clasificación de compases para el audio sintético ASR1.

tabla 4.1, la diferencia relativa en todos los casos da valores lejanos a 0, indicando que el poder de discriminación del sistema es bueno para este audio de prueba.

Pensando en observar de manera más amigable los resultados de la clasificación, se generó una aplicación que permite visualizar, compás a compás, un histograma de las verosimilitudes obtenidas. Además, se muestra la partitura de ese compás junto con la de los patrones usados para entrenar, de manera de comparar visualmente las diferencias o similitudes con cada uno. Esto permite analizar de manera más intuitiva los errores de clasificación que se obtengan, y permitirá intuir si se deben a un error de diseño en el sistema o simplemente a que el patrón a clasificar es similar a más de un patrón de entrenamiento. Además, puede ayudar a determinar las limitantes del enfoque elegido para la clasificación. En la figura 4.3 puede verse una captura del visualizador. En el programa también se marca con un recuadro verde la clase a la que pertenece el compás, y, en caso de que haya un error en la clasificación, con un recuadro rojo se marca la clase a la que fue erróneamente asignado.

Aquí se debe puntualizar que si se miran las log-verosimilitudes obtenidas para un mismo patrón en las distintas HMMs, esto no constituye una distribución de probabilidad, ya que están asociadas a procesos distintos (cada HMM tiene sus propios parámetros). De manera de transformar ese vector en algo parecido a una distribución, se realiza una normalización, que consiste simplemente en, para cada compás, dividir cada log-verosimilitud entre la suma de todas, obteniendo así un número entre 0 y 1. Matemáticamente, si se tienen p patrones a reconocer (es decir,

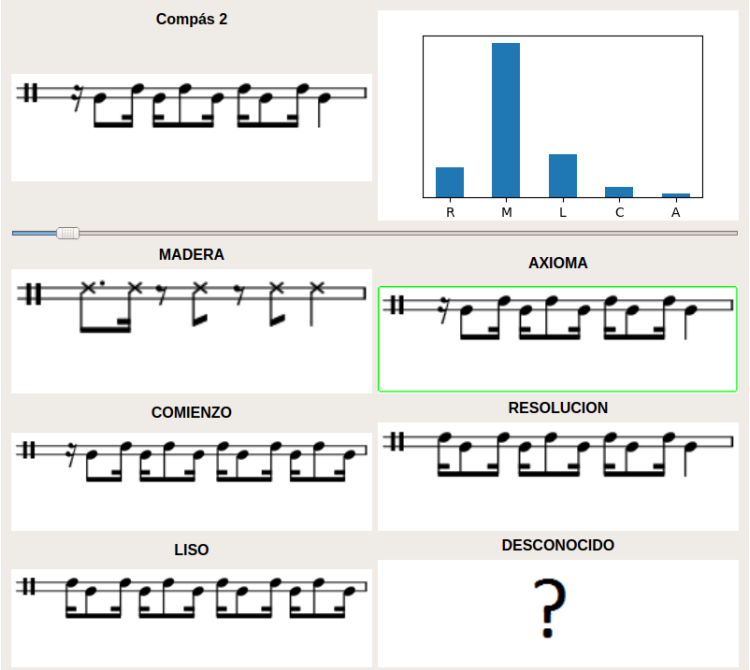


Figura 4.3: Visualizador de resultados de clasificación de compases.

p HMMs entrenadas) y $\mathbf{v}_c \in \mathbb{R}^p$ es el vector de log-verosimilitudes asociado a un compás c (con entradas $\mathbf{v}_c(i)$), el vector normalizado $\mathbf{n}_c$ se calcula:

$$\mathbf{n}_c = \frac{\mathbf{v}_c}{\sum_{i=1}^{p} \mathbf{v}_c(i)}. \tag{4.1}$$

Recordemos que la log-verosimilitud se obtiene a partir del algoritmo de Viterbi. Como se vio en el capítulo 2, en la práctica este algoritmo se implementa usando logaritmos, por lo que a la salida de la clasificación se obtiene el logaritmo de una probabilidad, lo que da siempre un número menor o igual a 0. Esto implica que las entradas de $\mathbf{v}_c$ son todas menores o iguales a 0, y por lo tanto, no es necesario usar valores absolutos en 4.1 para obtener un vector cuyas entradas tomen valores entre 0 y 1 y sumen 1.

Así, el histograma de la figura 4.3 se construye con el vector normalizado $\mathbf{n}_c$. Por la naturaleza de la normalización y de las cantidades involucradas, la clase que obtiene una mayor log-verosimilitud es la asociada a una barra con altura más pequeña en el histograma. Esto se debe a que las log-verosimilitudes toman valores negativos, y se selecciona la mayor, es decir, la más cercana a 0 (recordemos que la log-verosimilitud es el logaritmo de una probabilidad, por lo que cuanto más cercana a 0 sea, más cercana a 1 es la probabilidad). Al normalizar, el resto de las log-verosimilitudes son más negativas, por lo que al dividir cada valor entre la suma de todas se transforma la log-verosimilitud más alta (la más cercana a 0) en el valor más bajo. Un ejemplo numérico: para el compás que se muestra en la figura 4.3, el vector de log-verosimilitudes obtenido es

$\mathbf{v}_c = [-4481.88, -23012.90, -6440.44, -1620.65, -628.81]$ (las log-verosimilitudes se corresponden a las clases *R, M, L, C* y *A* respectivamente, el mismo orden en el que aparecen en el histograma del visualizador). Si se realiza la normalización dada por la ecuación 4.1, se obtiene $\mathbf{n}_c = [0.124, 0.636, 0.178, 0.045, 0.017]$ (el resultado se obtiene simplemente dividiendo cada entrada de $\mathbf{v}_c$ entre -36184.69, que es la suma de todas sus entradas). Así, la clase que obtuvo mayor log-verosimilitud (la *A*, que es a la que realmente pertenece el compás) es la asociada al menor valor del vector $\mathbf{n}_c$ (y así se refleja en el histograma de la figura 4.3).

Para la prueba usando el audio sintético ASR2, los resultados se presentan en la tabla 4.2.

Compás	*Ground truth*	*Clasificación (score)*	*Siguiente máximo (score)*	*Dif. relativa*
1	M (original)	M (-1889.66)	C (-16558.13)	0.886
2	M (modificado)	M (-2433.95)	A (-9892.21)	0.754
3	M (modificado)	M (-2673.62)	A (-10840.67)	0.753
4	A (original)	C (-1877.85)	A (-2036.55)	0.078
5	A (sustitucion 4.1)	A (-2332.67)	C (-3519.79)	0.337
6	A (sustitucion 4.1)	A (-2253.04)	C (-3602.63)	0.375
7	A (sustitucion 4.1)	A (-2298.53)	C (-3837.76)	0.401
8	A (densificacion 3.1)	A (-371.49)	C (-1785.08)	0.792
9	A (densificacion 3.2.1)	A (-698.48)	C (-1877.83)	0.628
10	A (densificacion 3.1+3.2.1)	A (-1042.10)	C (-2208.81)	0.528
11	A (densificacion 3.2.3)	A (-248.66)	C (-1524.90)	0.837
12	A (densificacion 3.1+3.2.3)	A (-440.45)	C (-2034.10)	0.783
13	C (original)	C (-528.10)	A (-802.27)	0.342
14	L (densificacion 3.2.1)	L (-302.10)	R (-946.07)	0.681
15	R (original)	R (-447.93)	L (-1513.50)	0.704
16	A (original)	A (-245.21)	C (-1902.87)	0.871
17	C (densifiacion 3.1)	C (-625.34)	A (-945.90)	0.339
18	L (densificacion 3.2.3)	L (-278.86)	R (-767.72)	0.637
19	L (densificacion 3.2.1)	L (-276.56)	C (-850.20)	0.675
20	L (densificacion 3.2.3+3.2.3)	L (-516.23)	R (-705.69)	0.268
21	L (densificacion 3.2.1+3.2.3)	L (-456.74)	R (-1029.00)	0.556
22	R (densificacion 3.1)	R (-353.63)	L (-1618.88)	0.782
23	A (sustitucion 4.2)	A (-795.84)	C (-2473.80)	0.678
24	C (sustitucion 4.3)	A (-1454.51)	C (-1641.63)	0.114
25	R (sustitucion 4.2)	A (-1824.91)	R (-2101.72)	0.132

Tabla 4.2: Resultados de la clasificación de compases para el audio sintético ASR2. En gris se resaltan los compases mal clasificados.

Esta tabla es la equivalente a la presentada para la primera prueba, con la diferencia que en la segunda columna, además de indicar a qué clase pertenece el compás, se indica entre paréntesis si está modificado y qué tipo de modificación es.[4]

[4]Los números que allí aparecen hacen referencia a la sección del trabajo de Jure [34]

Respecto a los resultados obtenidos, lo primero a destacar es que solo las compases 4, 24 y 25 con clasificados incorrectamente; el resto es asignado a la clase correcta. Si se mira la diferencia relativa para esos compases, se observa que son los que obtuvieron valores más cercanos a 0, lo que indica un bajo nivel de confianza en la clasificación, y refuerza la idea de que esta medida es útil para tener una noción de la confiabilidad del sistema al clasificar un compás.

Un hecho a resaltar en el caso de los compases que son incorrectamente clasificados es que la clase que obtiene el segundo máximo es siempre la clase correcta, y con una log-verosimilitud muy cercana a la del primer máximo. Por ejemplo, el compás 24 es un C modificado, y, si bien es clasificado como un A, la clase C obtiene el segundo puntaje máximo, con una diferencia relativa de 0.114. Algo similar sucede con los compases 4 y 25.

El vigésimoquinto compás es interesante pues muestra además una limitante del enfoque elegido para clasificar compases. Si se observa la partitura para los compases 23, 24 y 25 (que se muestra ampliada en la figura 4.4), se verá que los compases 23 y 25 son iguales. Pero, si se observa la segunda columna de la tabla 4.2, resulta que según el *ground-truth* el compás 23 es un A, mientras que el 25 es un R.

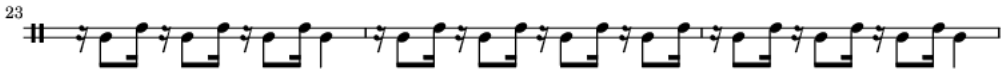


Figura 4.4: Partitura de los compases 23, 24 y 25 del audio ASR2.

Para explicar esta diferencia, debemos volver a las reglas de Jure. La transformación aplicada al compás 23 es la sustitución, en el axioma, del núcleo N por una sucesión de dos inicios I-I, como en la figura 4.5. De ahí que ese compás se considere una modificación del axioma.

Figura 4.5: Regla de sustitución de un N por una sucesión I-I. Figura extraída de [34].

Debido a que la sucesión I-I toma el lugar de N, hereda su regla de expansión, como se muestra en la figura 4.6. Esto es lo que sucede en el compás 24 de 4.4.

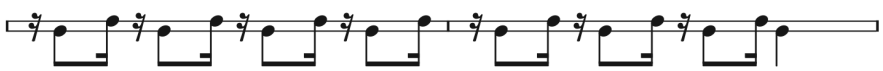


Figura 4.6: Expansión de la sucesión I-I, que ocupa el lugar del núcleo N. Figura extrída de [34].

donde se introduce(n) la(s) modificación(es) presente(s) en ese compás.

Por lo tanto, el compás 24 puede pensarse como un comienzo C, en el cual cada pulso del núcleo es sustituido por una I, y el 25 como la resolución de la regla de expansión empezada en el compás 24. Una combinación de estas reglas se muestra gráficamente en la figura 4.7. Allí se ve además que la combinación de esta regla de sustitución con la expansión lleva a otra ambigüedad: el segundo compás es una modifiación de un liso (por lo que deberá ser considerado una variante de la clase L), pero resulta igual al primer compás, que es el comienzo de la regla de expansión, luego de la sustitución de I-I por N.

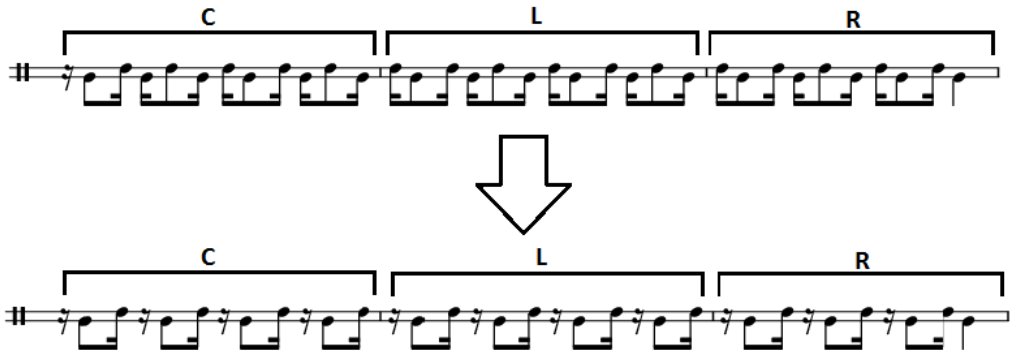


Figura 4.7: Combinación de reglas de sustitución y expansión que causa ambigüedades en la asignación de clases.

De ahí que, si bien los compases 23 y 25 son iguales, deban ser considerados como pertenecientes a clases diferentes. Esto muestra que el sistema no está diseñado para absorber estas modificaciones, por lo que los errores de clasificación en esos casos son razonables.

Respecto al resto de los compases, se pueden destacar varias cosas. Comencemos con los compases 5 a 7. Si se vuelve a la partitura 4.2, se observará que estos compases son bastante distintos a los patrones de entrenamiento, especialmente en términos de cantidad de golpes en un compás; para ver esto de manera más clara, en la figura 4.8 se muestra el visualizador de clasificación para el quinto compás.

Es interesante notar que la clasificación es correcta a pesar de las claras diferencias de ese compás con los patrones de entrenamiento. Como se vio en el capítulo 1.2, esta modificación, que consiste en la sustitución del núcleo N por la sucesión F-I, solo puede ser realizada dentro del axioma según las reglas de Jure[5]. De ahí que este patrón sea considerado (y deba ser clasificado) como una variación de A. Destacable es también que, si además de esta transformación se aplica además un adorno del I como el agregado de un golpe de palo, el sistema reconoce de igual manera ese compás como una variante del axioma. Ejemplo de ello son los compases 6 y 7 de 4.2; la figura 4.9 muestra el visualizador para el sexto compás,

[5]Dado que esta sutitución solo puede hacerse dentro del axioma, en este caso no hay riesgo de introducir ambigüedades como sucedía antes: estos compases siempre serán considerados una variante del axioma.

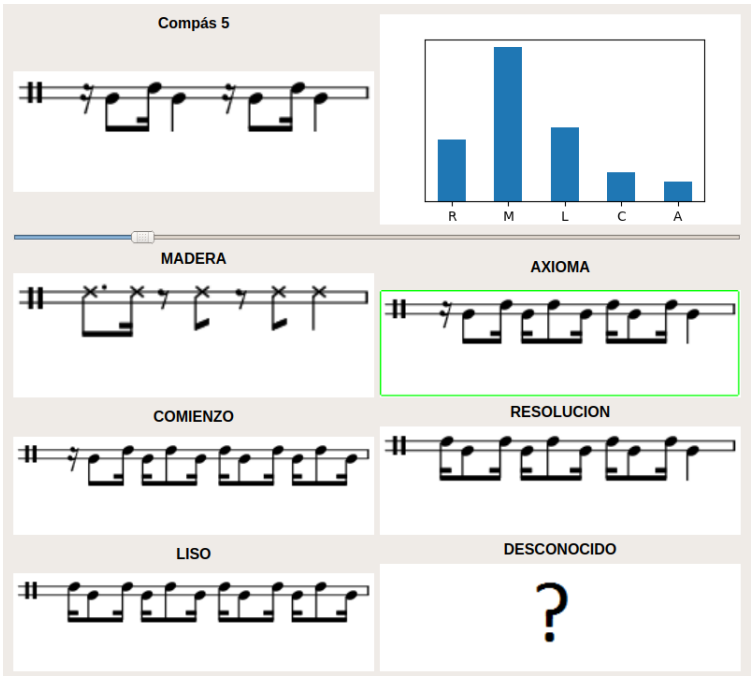


Figura 4.8: Resultados de clasificación del quinto compás, para el audio ASR2.

donde solo uno de los I está adornado. Si ambos se adornan, como en el compás 7, la clasificación es correcta de todos modos.

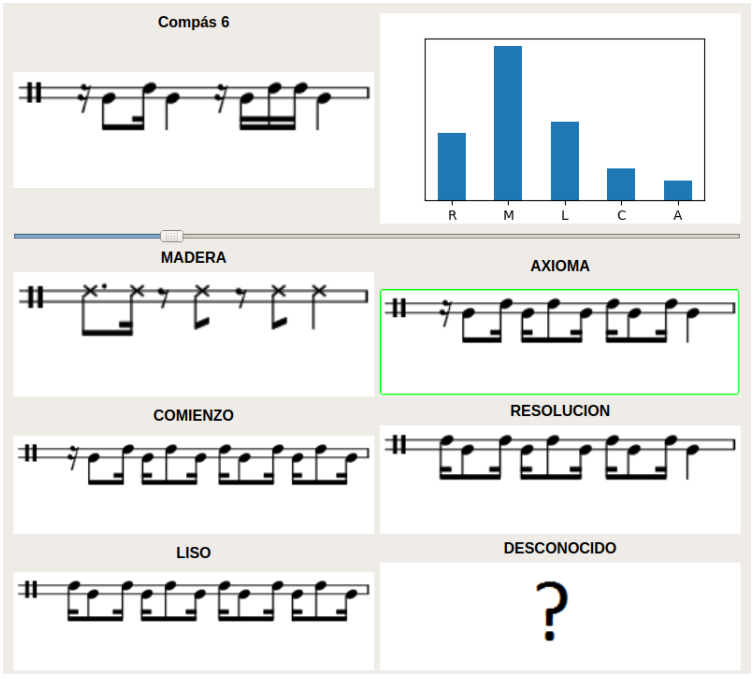


Figura 4.9: Resultados de clasificación del sexto compás, para el audio ASR2.

Otros casos de adornos que resultan en compases más similares a los patrones originales también son bien clasificados. Por ejemplo, el compás número 20 de 4.2 es un ejemplo de patrón L donde se agrega un golpe de palo en el segundo y el cuarto pulso. La figura 4.10 muestra el resultado de la clasificación en ese caso. Si bien en el histograma se observa que las clases R y L obtienen verosimilitudes similares, el mínimo se da en la L, que es la clase correcta.

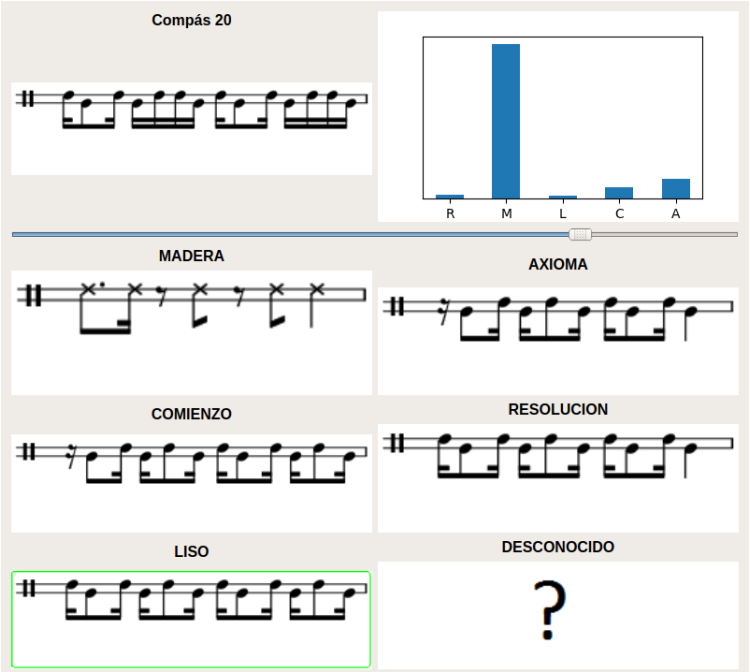


Figura 4.10: Resultados de clasificación del vigésimo compás, para el audio ASR2.

De forma similar, el compás 18 es una variante de L, pero a diferencia del 20, solo se agrega un golpe de palo en el cuarto pulso, como se ve en la figura 4.11. En ese caso la clasificación también es correcta, pero la diferencia relativa entre el primer y el segundo máximo de la log-verosimilitud es mayor que para el vigésimo compás (0.637 para el 18 frente a 0.268 en el 20).

Otro aspecto a destacar, que es patente en las figuras 4.8, 4.9, 4.10 y 4.11, es que la madera siempre obtiene una verosimilitud muy diferente del resto de las clases en los patrones que no son madera, lo que indica que el sistema tiene en cuenta el contenido tímbrico de la señal. Esto es una consecuencia esperable de usar los MFCCs como características. Un ejemplo de clasificación de un compás de madera se muestra en la figura 4.12, donde puede apreciarse la sensible diferencia entre la verosimilitud de la clase madera y las del resto. Si bien el compás a clasificar es una modificación de la clave, el sistema la reconoce claramente como un compás de madera.

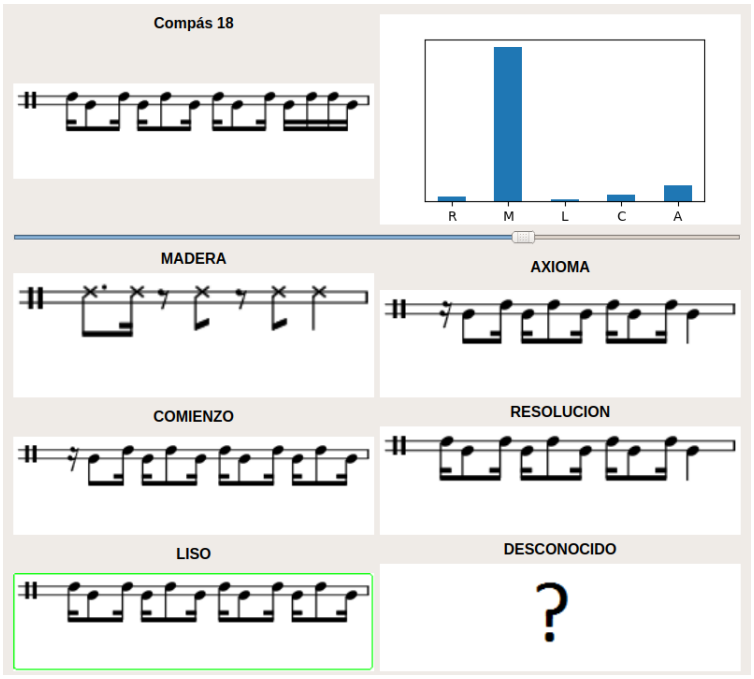


Figura 4.11: Resultados de clasificación del decimoctavo compás, para el audio ASR2.

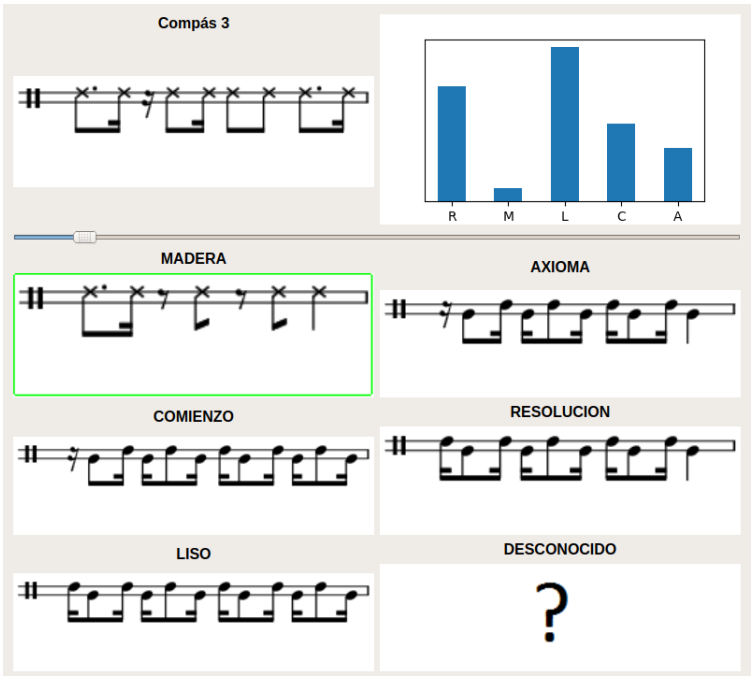


Figura 4.12: Resultados de clasificación del tercer compás, para el audio ASR2.

Clasificación de pulsos

Para la clasificación de pulsos, recordemos que existen cuatro clases a las que los patrones se van a asignar: inicio (I), núcleo 1 (N_1), núcleo 2 (N_2) y final

(F), según la nomenclatura introducida en la sección 1.2.[6] Los resultados que se presentan a continuación provienen de intentar asignar a estas cuatro clases cada pulso de los audios sintéticos ASR1 y ASR2.

Comencemos con el audio ASR1. En ese caso, el porcentaje de clasificación de pulsos obtenido fue de un 100 %, es decir, los 64 pulsos se clasifican correctamente.[7] Cabe señalar que los patrones N_1 y N_2 solo se diferencian en que los golpes de mano y palo están intercambiados. Por esta razón, que el sistema detecte correctamente cada patrón indica que está teniendo en cuenta el contenido tímbrico de la señal.

En la figura 4.13 se muestra un gráfico de barras de la diferencia relativa entre el primer y el segundo máximo de la log-verosimilitud para la clasificación de cada pulso.

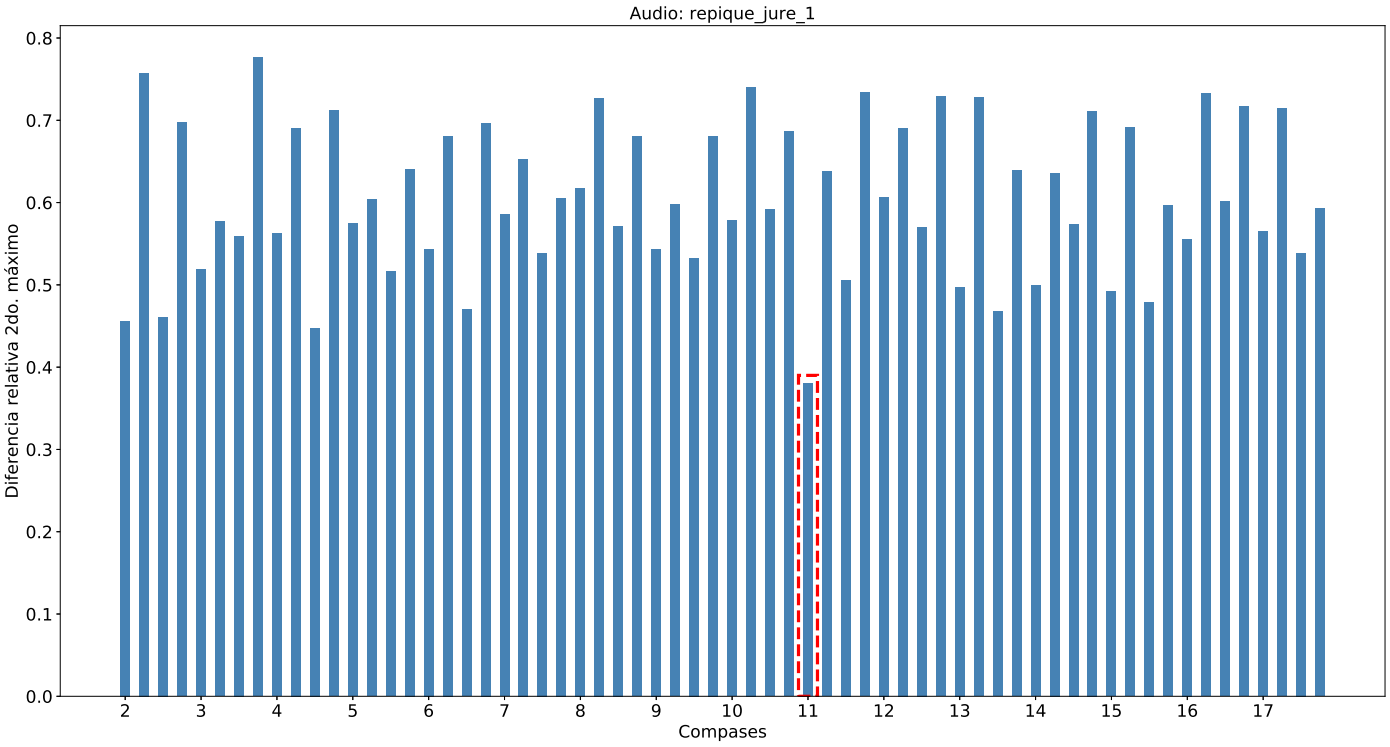


Figura 4.13: Diferencia relativa entre los primeros dos máximos de la log-verosimilitud, para la clasificación de pulsos del audio ASR1. En rojo se indica el pulso que obtiene la menor diferencia.

En esa figura puede observarse que la diferencia relativa en general es lejana a 0. El pulso que resulta con una confianza más baja en la clasificación es el primer pulso del compás 11. Para analizar en más detalle qué pasa en esta situación, podemos recurrir al visualizador que se utilizó antes para los compases, que fue adaptado para visualizar la clasificación de pulsos.

[6] Vale recordar también que en este caso no se clasifica la madera.

[7] La pieza está compuesta por 17 compases, y el primero es de madera, por lo que se tienen $16 \times 4 = 64$ pulsos a clasificar.

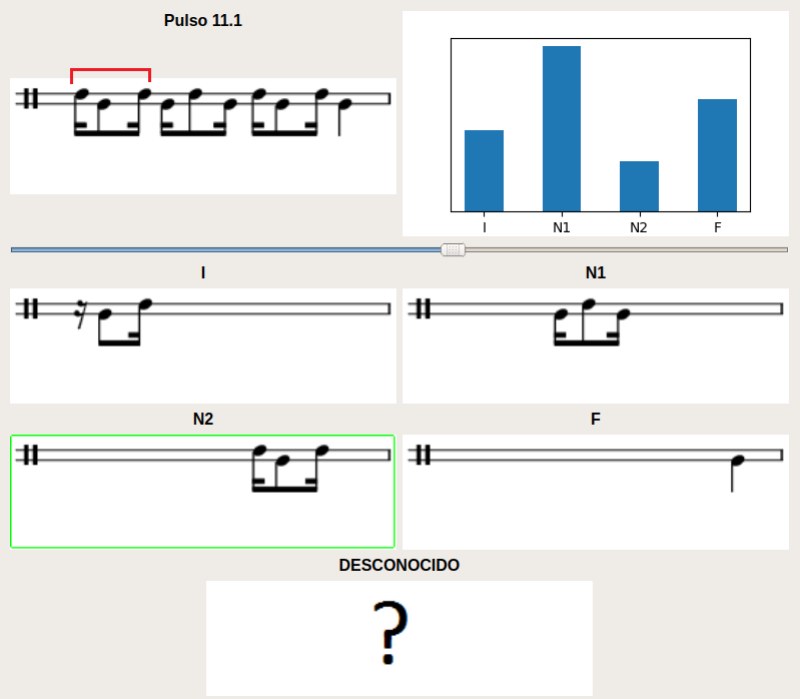


Figura 4.14: Visualizador de clasificación de pulsos para el primer pulso del onceavo compás del audio ASR1.

En la figura 4.14 se muestra el visualizador para el primer pulso del onceavo compás. Allí se ve que el histograma para este pulso alcanza su mínimo en la clase correcta (la N_2), y esa cantidad es sensiblemente menor que la asociada a las otras clases (recordemos que por la normalización realizada, la clase con mayor log-verosimilitud está asociada a la menor altura en las barras del histograma).

Si queremos analizar con más detalle qué es lo que sucede, en el visualizador es posible mirar los estados ocultos que recorre cada cadena dadas las observaciones en cada pulso del audio de entrenamiento. La sucesión de estados ocultos es estimada a partir de la secuencia de observaciones con el algoritmo de Viterbi, según lo explicado en la sección 2.2.2. Para el pulso en cuestión, esa estimación se muestra en la figura 4.15.

En esa figura se muestra, para cada trama de audio, el estado oculto en el que cada HMM se encuentra según el algoritmo de Viterbi. También se ve allí la ubicación temporal de los golpes (en una línea punteada roja). Como se explicó en la sección 3.1.3, cada cadena se modela como una sucesión de golpes y silencios, por lo que es de esperar que, donde hayan golpes, la cadena se encuentre en estados asociados a golpes. Ejemplifiquemos con el caso de un pulso, por ejemplo de la clase I. Este patrón consiste en un golpe de palo en el segundo *tatum* del pulso, y un golpe de palo en el cuarto, como se ve por ejemplo en la figura 4.5. Así, en el caso ideal, la cadena debería comenzar en un silencio[8], pasar algunos instantes

[8]Recordemos que, como fue mencionado en el Capítulo 3, se agrega un tiempo al inicio

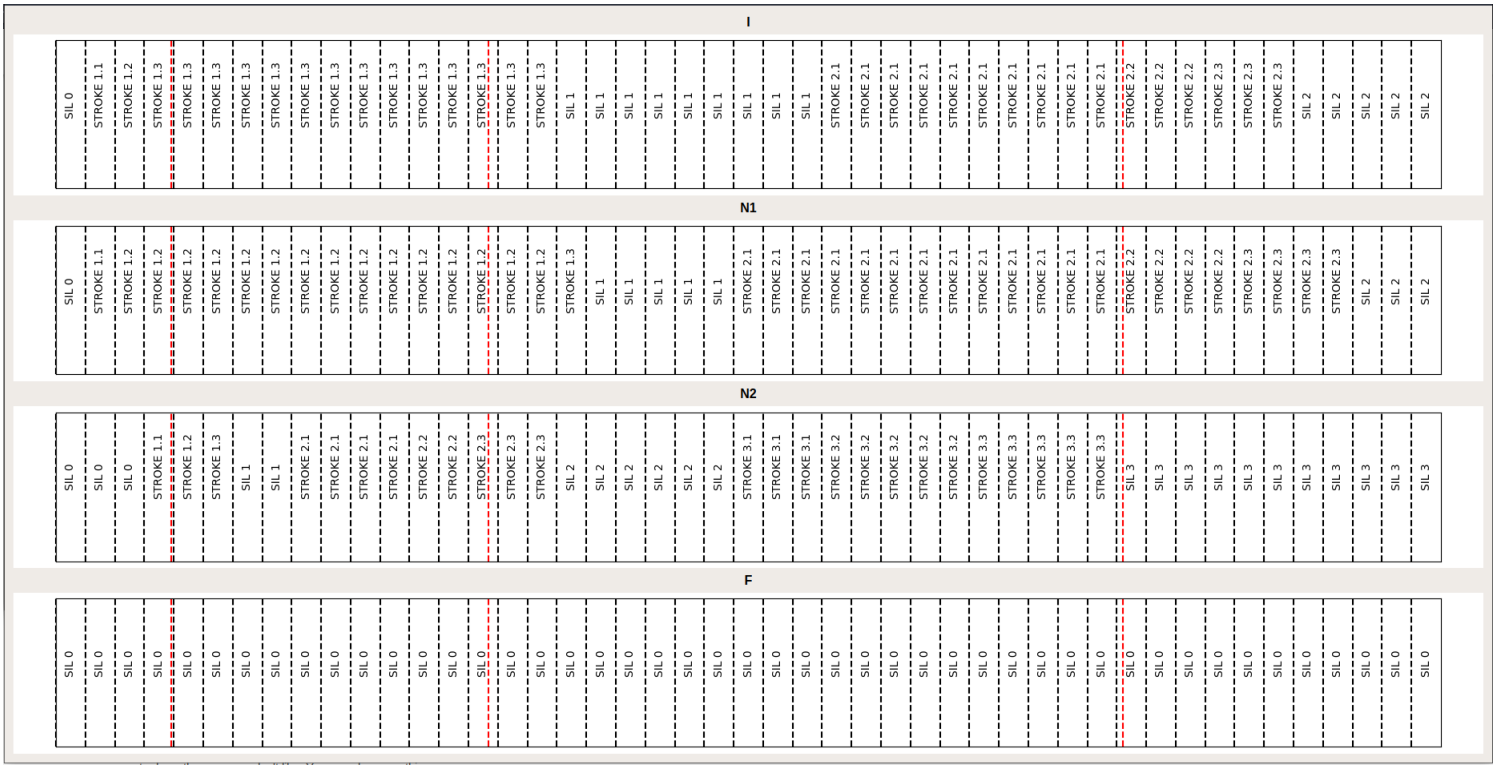


Figura 4.15: Secuencia de estados ocultos, según las distintas HMMs, para el primer pulso del onceavo compás del audio ASR1.

de tiempo allí, para de ahí saltar a un estado asociado a un golpe. Allí pasaría algunos otros instantes (recorriendo los diferentes sub-estados asociados al golpe), para volver a saltar a un silencio. Y luego seguiría su recorrido, obteniendo algo similar a lo que ya vimos en la figura 3.3 del Capítulo 3, cuando se explicó cómo se etiquetaban los estados ocultos.

La idea de la figura 4.15 es relevar si la cadena está haciendo un recorrido similar a este, o si en cambio la sucesión de estados ocultos visitados no se condice con la ubicación temporal de los golpes. En esa figura se muestran los estados ocultos estimados no solo para la cadena correcta, si no que también se calculan para el resto de las HMMs, de manera de analizar qué es lo que el sistema está capturando a partir de las características observadas. Así, por ejemplo vemos que la cadena asociada al patrón F se queda siempre en el estado inicial de silencio, y nunca se mueve de ahí a pesar de que se suceden los golpes. Esto es razonable, ya que el pulso que estamos intentando clasificar no se parece en nada a los que esa cadena está entrenada para reconocer, y por lo tanto la secuencia de estados que se obtiene no es representativa de lo que en realidad sucede.

Si se vuelve al histograma de la figura 4.14, se observará que para ese pulso las clases más probables son (en orden) la N_2 y la I. Empecemos por analizar la

de cada pulso de manera de asegurar que todas las cadenas comiencen en un estado de silencio

secuencia de estados ocultos asociados a la cadena I, que se ve en la primera fila de 4.15. Lo que observamos es que la cadena comienza en un estado inicial de silencio, para saltar enseguida a un estado asociado a un golpe, lo que es coherente ya que a los pocos instantes de tiempo hay efectivamente un golpe (el primer golpe de palo del pulso). Lo interesante sucede a continuación: la cadena se mantiene en ese estado hasta que pasa el siguiente golpe (este de mano), y ahí salta a un estado de silencio. Luego, vuelve a pasar a un estado asociado a un golpe, y se mantiene allí hasta que sucede el último golpe de palo. Esto es razonable si se piensa en el tipo de patrón que esta cadena está entrenado para reconocer: el patrón I solo tiene dos golpes, por lo que, al haber un golpe de más en este caso, la cadena busca asimilarlo a alguno de los sub-estados asociados a los otros golpes (en este caso, agrupa los primeros dos en el tercer sub-estado del primer golpe). Si además observamos que un pulso N_1 solo se diferencia de un I en el primer golpe de palo, este recorrido resulta aún más verosímil todavía.

En el caso de la cadena N_2, observamos que el recorrido es más o menos el esperado: la cadena recorre los estados ocultos de manera tal que todos los golpes quedan dentro de estados que se corresponden con la presencia de un golpe, y con el número de golpe en el fragmento de audio. Es decir, el primer golpe cae dentro de un conjunto de estados que la cadena asocia con el primer golpe, y el segundo también. Con el tercero lo que sucede es que cae apenas afuera de la secuencia de estados que la cadena asocia al tercer golpe. Es probablemente debido a este pequeño desfasaje que la log-verosimilitud es un poco mayor en este caso particular, haciendo que la diferencia relativa con el siguiente máximo baje un poco.

La siguiente prueba, al igual que para la clasficación de compases, se realizó sobre el audio ASR2. En ese caso, el porcentaje de acierto en la clasificación fue también de un 100 %. Esto implica que todos los 88 pulsos de la pieza se clasifican correctamente.[9] En la figura 4.16 se muestra la diferencia relativa entre los primeros dos máximos de la log-verosimilitud. Nuevamente, se observa que la mayoría de las clasificaciones tiene un buen nivel de confiabilidad según esta medida.

Lo interesante que surge a partir de aquí es cuáles son los patrones que tienen baja confiabilidad en la clasificación. Si analizamos la figura 4.16, vemos que los pulsos con menores niveles de confiabilidad son: el segundo pulso del compás 9, el cuarto del 18, y el segundo del 22. Si se vuelve a la partitura de este audio en la figura 4.2, o se observan estos pulsos con el visualizador como en la figura 4.17, se constatará que en los tres casos se trata de patrones adornados.

Así, el sistema no solo es capaz de asignar los pulsos adornados a la clase correcta, sino que también da una medida que puede indicar si se trata o no de una variante. Por ejemplo, podríamos definir un umbral para la diferencia relativa, y clasificar como variante a aquellos pulsos que caigan por debajo de este umbral; en la figura 4.18 se muestra el resultado que se obtendría si ese umbral se fijase en 0.4.

En la tabla 4.3 se muestran los pulsos que se clasifican como variantes al

[9] El audio tiene 25 compases, de los que los tres primeros son de madera. Como para la clasificación de pulsos esos no se utilizan, quedan 22 compases (u 88 pulsos) para clasificar.

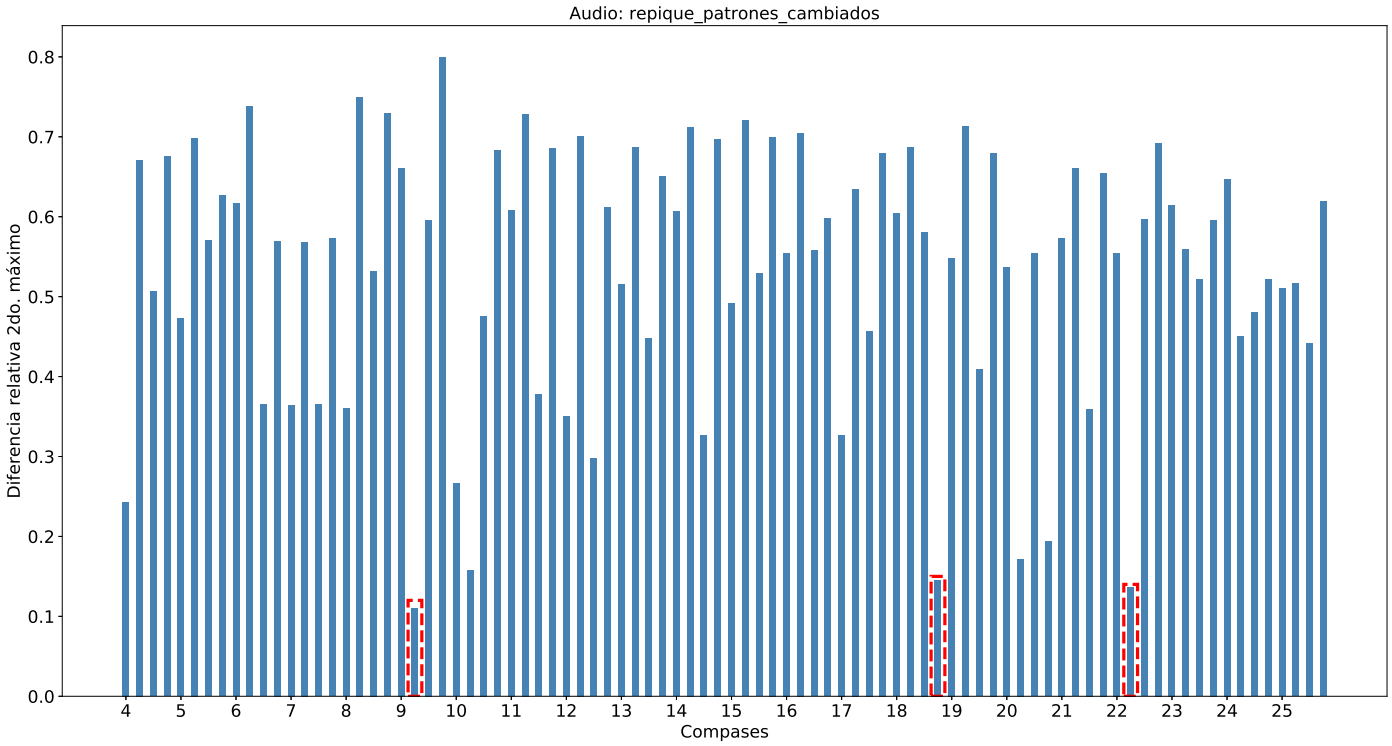


Figura 4.16: Diferencia relativa entre los primeros dos máximos de la log-verosimilitud, para la clasificación de pulsos del audio ASR2. En rojo se indican los pulsos que obtienen menores diferencias.

seguir ese camino. De todos ellos, el único que se clasifica como variante cuando en realidad no lo es es el primer pulso del cuarto compás. Es decir, 17 de los 18 pulsos se identifican correctamente como variaciones.[10] Si se vuelve a la partitura de la figura 4.2, se observará que solo un pulso que se encuentra adornado no se clasifica como variación: es el tercer pulso del compás 19. En el histograma 4.18 se puede ver que ese pulso apenas logra superar el umbral de distancia relativa, por lo que de haber sido este apenas superior esa variante hubiese sido captada. De todas maneras, no se pretende aquí definir una manera óptima de encontrar las variaciones, o ni siquiera de determinar el umbral óptimo, si no que la idea es mostrar que con este enfoque es posible clasificar correctamente los patrones y detectar cuándo se encuentran adornados.

[10]De nuevo, vale la pena aclarar que no solo se identifica que el pulso es una variación, si no que también se dice de qué patrón es una variación.

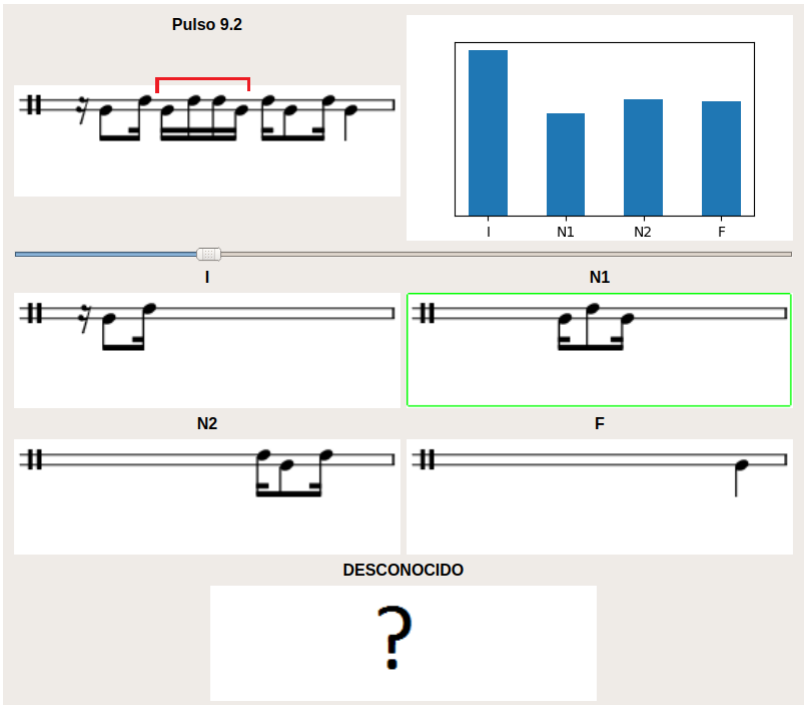


Figura 4.17: Visualizador de clasificación para el segundo pulso del noveno compás, para el audio ASR2.

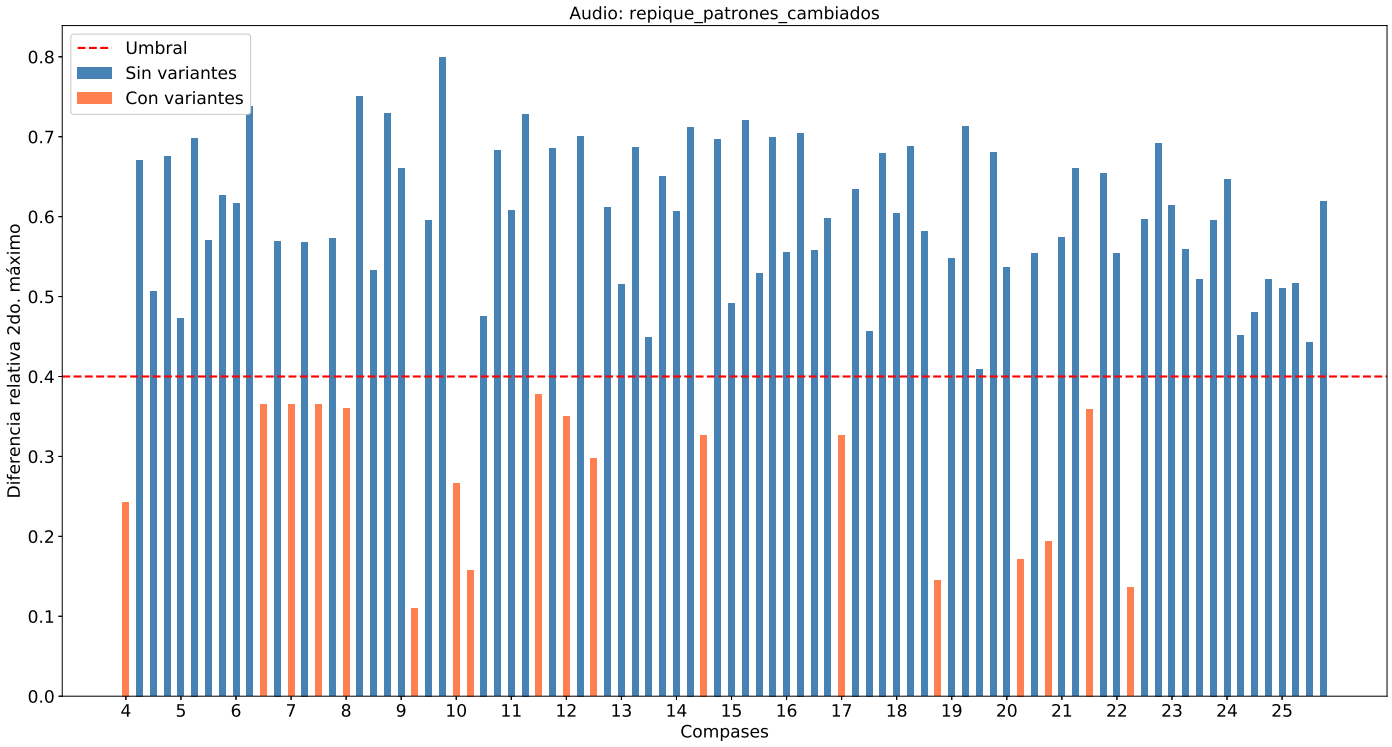


Figura 4.18: Resultado de umbralizar el valor de la distancia relativa para determinar si los pulsos presentan variantes, audio ASR2.

Pulso	*Clasificación*	*¿Correcto?*
4.1	I	No
6.3	I	Si
7.1	I	Si
7.3	I	Si
8.1	I	Si
9.2	N_1	Si
10.1	I	Si
10.2	N_1	Si
11.3	N_2	Si
12.1	I	Si
12.3	N_2	Si
14.3	N_2	Si
17.1	I	Si
18.4	N_1	Si
20.2	N_1	Si
20.4	N_1	Si
21.3	N_2	Si
22.2	N_1	Si

Tabla 4.3: Pulsos clasificados como variantes, audio ASR2. La notación $x.y$ indica el pulso número y del compás x. Por ejemplo, 6.3 indica el tercer pulso del sexto compás.

4.1.2. Entrenamiento con audio sintético y clasificación con audio real

Hasta este punto, solo se ha trabajado con audio sintético, tanto para el entrenamiento como para la clasificación. Surge entonces como continuación natural el uso de grabaciones reales. Dada la falta de datos sobre la que ya se ha insistido bastante, no es posible realizar el entrenamiento con audio real, pero ¿qué pasa si intentamos clasificar audios reales usando las cadenas entrenadas con audio sintético?

Para responder esta pregunta, analizaremos el desempeño del sistema al intentar clasificar el audio transcripto en la figura 4.19 que, siguiendo con la nomenclatura que venimos utilizando, denominaremos ARR1. Este audio, correspondiente a un fragmento de una grabación real,[11] tiene algunos compases similares a los patrones de entrenamiento, mientras que otros se alejan bastante de los mismos. En ese sentido, es una buena opción para evaluar la capacidad del sistema entrenado con audio sintético para adaptarse al caso real. El intérprete en este caso es Héctor Manuel Suárez, músico de candombe conocido como un virtuoso del repique y del piano, y que proviene del estilo del barrio Palermo.

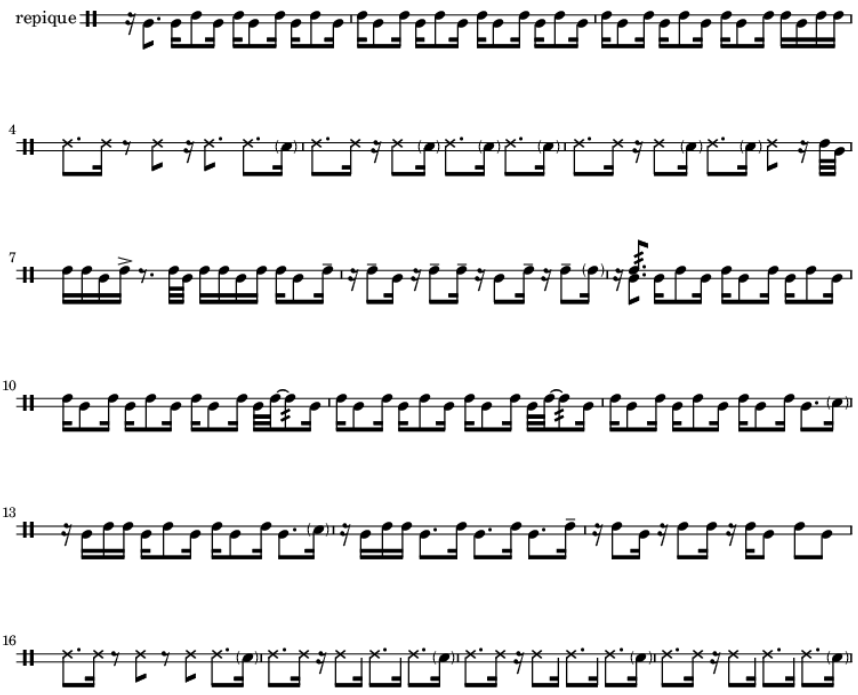


Figura 4.19: Partitura del audio ARR1 usado para clasificación de variantes de repique.

Se debe hacer una aclaración para este audio: a diferencia de las pruebas del caso sintético, en las que los audios a reconocer estaban compuestos o por los

[11] Por más detalles sobre la sesión de grabación en la que fue registrado ese audio, ver [61].

patrones de entrenamiento o por variantes de ellos, aquí hay patrones que las cadenas nunca vieron (por ejemplo, en los compases 7, 8, 14 y 15). Esos patrones serán considerados "desconocidos", y se espera que el sistema tenga una baja confiabilidad al intentar clasificarlos.

Hecha la aclaración, se describen a continuación los resultados obtenidos al clasificar este audio, para las dos unidades a reconocer: compases y pulsos.

Clasificación de compases

Para la clasificación a nivel de compases, los resultados se muestran en la tabla 4.4. Como allí puede verse, el desempeño es muy pobre en este caso. Si bien dos compases son clasificados correctamente (el 1 y el 13), las clasificaciones obtenidas indican que el sistema no está siendo capaz de adaptarse al caso real. Esto queda especialmente patente si se mira la diferencia relativa entre los dos primeros máximos de la log-verosimilitud: en todos los compases esta cantidad es muy pequeña, indicando un bajo nivel de confianza en la clasificación.

Compás	*Ground truth*	*Clasificación (score)*	*Siguiente máximo (score)*	*Dif. relativa*
1	C	C (-9816.91)	A (-11206.71)	0.124
2	L	A (-10120.16)	C (-10419.46)	0.029
3	L	A (-10814.49)	C (-11371.12)	0.049
4	M	A (-7703.16)	C (-10035.41)	0.232
5	M	A (-9488.00)	C (-11093.02)	0.145
6	M	A (-9203.83)	C (-11129.48)	0.173
7	Desconocido	A (-12856.02)	C (-14206.79)	0.095
8	Desconocido	A (-12777.81)	C (-14285.63)	0.106
9	C	A (-9108.24)	C (-10199.31)	0.107
10	L	A (-10115.61)	C (-10582.28)	0.044
11	L	C (-11486.60)	A (-12631.07)	0.091
12	R	C (-12573.80)	A (-13230.20)	0.050
13	A	A (-9333.28)	C (-10395.83)	0.102
14	Desconocido	A (-10509.39)	C (-11240.00)	0.065
15	Desconocido	A (-12231.35)	C (-12384.79)	0.012
16	M	A (-10195.37)	C (-11238.62)	0.093
17	M	A (-9770.94)	C (-11196.22)	0.127
18	M	A (-9722.33)	C (-10248.83)	0.051
19	M	C (-9914.45)	R (-11671.41)	0.151

Tabla 4.4: Resultados de la clasificación de compases para el audio ARR1.

Otro indicador claro del pobre desempeño en este caso es lo que sucede con los compases de madera. Para audios sintéticos, esos compases siempre eran clasificados correctamente, y en ellos se obtenían los mayores índices de confiabilidad. Ahora, no solo esos compases no son clasificados correctamente, si no que la clase

M nunca aparece entre los dos primeros máximos de la log-verosimilitud. Si por ejemplo analizamos el cuarto compás con el visualizador, como en la figura 4.20, observamos que la clase M aparece recién como cuarta opción de clasificación. Esto quiere decir que, si se observa el histograma de 4.20, la M es la clase que obtiene la cuarta menor log-verosimilitud, por lo que el sistema no reconoce este patrón ni siquiera como remotamente cercano a una madera.

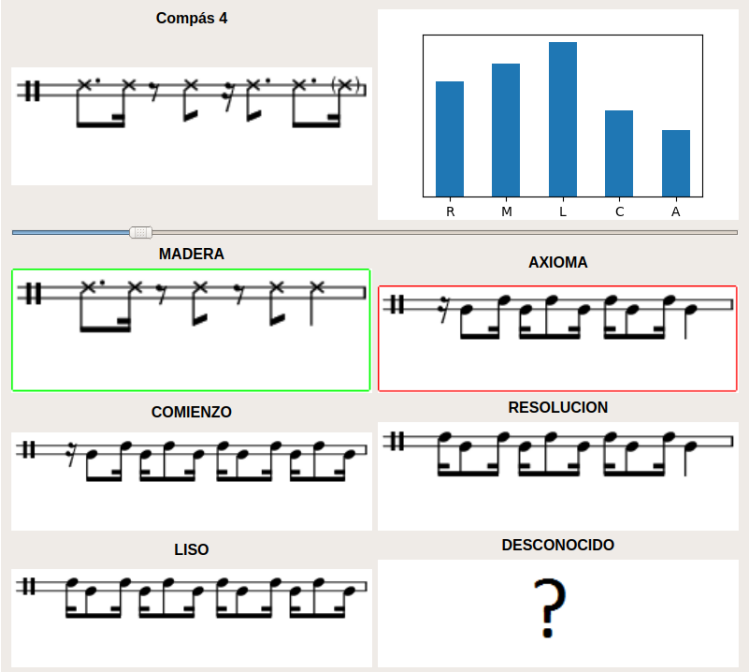


Figura 4.20: Visualizador de clasificación para el cuarto compás, para el audio ARR1.

Si se vuelve a la tabla 4.4, se verá también que todos los compases con clasificados dentro de las mismas dos clases: o C o A, y en casi todos los compases estas clases se alternan entre los dos primeros máximos de la log-verosimilitud.

Por lo tanto, todo apunta a que el sistema no es capaz de generalizar bien para el caso de audios reales. Una posible explicación de este comportamiento es la diferencia entre las características acústicas de los audios usados para entrenar (sintéticos) y el que se está intentando clasificar. Si bien la grabación fue registrada en condiciones muy favorables (en una sala, con canales independientes para los tambores para minimizar el ruido cruzado), la situación de fondo sigue siendo comparar un audio real con uno sintético. Nunca queda más clara la diferencia entre un audio real y un audio sintético que al compararlos auditivamente; se insta a que el lector lo haga con los audios de este trabajo.[12]

En trabajos de reconocimiento de voz hablada que usan HMMs (como [23, 40, 43]), es usual realizar algún tipo de adaptación acústica al momento de clasificar a

[12]En el apéndice A se indica dónde de encuentran los audios dentro de los archivos entregados con la tesis para que el lector pueda escucharlos. Se puede leer allí también sobre cómo ejecutar las pruebas que aquí se reportan.

un nuevo hablante, pues se parte de la premisa de que el sistema de reconocimiento fue entrenado con diferentes hablantes. Se intenta entonces pasar de un sistema "independiente del hablante" a uno "dependiente". Esto quiere decir que se asume que las propiedades acústicas de la señal que se intenta clasificar difieren con las del conjunto de entrenamiento, y se busca modificar los parámetros de las HMMs para que modelen mejor las propiedades acústicas de la voz a reconocer. En [55], trabajo que usa HMMs para detectar los distintos sonidos de batería y por lo tanto tiene características similares a esta tesis, este enfoque también es utilizado. Dados los resultados obtenidos hasta ahora para los audios reales, resulta entonces razonable intentar realizar algún tipo de adaptación para ver cómo se modifica el desempeño del sistema.

La idea es modificar los parámetros de las HMMs para de alguna manera mejorar el modelado del sistema sobre el audio real. El enfoque utilizado aquí será el de realizar una adaptación acústica con regresión lineal de máxima verosimilitud, como la descripta en [40]. En particular, se buscará modificar las medias de las normales que modelan las observaciones en cada estado oculto. Como se recordará de lo visto en la sección 3.1.5, las distribuciones de probabilidad de las observaciones para cada estado se asumen gaussianas en $\mathbb{R}^{10}$, con matriz de covarianza diagonal (es decir, se supone que las coordenadas del vector de observaciones son independientes). Así, si llamamos $\mu_s \in \mathbb{R}^{10}$ a la media de la normal asociada al estado s, el vector adpatado $\hat{\mu_s}$ se calculará aplicando una matriz de transformación W_s al vector de medias extendido ξ_s:

$$\hat{\mu_s} = W_s \xi_s \tag{4.2}$$

donde W_s es una matriz 10×11 y $\xi_s = [\omega, \mu_s(1), \mu_s(2), \ldots, \mu_s(10)]^t$. El parámetro ω puede valer 0 o 1, y se utiliza para introducir un offset en la regresión.[13] Dado que las coordenadas del vector de observaciones se asumen independientes, buscaremos aquí una matriz de transformación de la forma:

$$W_s = \begin{pmatrix} w_{1,1} & w_{1,2} & 0 & \ldots & 0 \\ w_{2,1} & 0 & w_{2,3} & \ldots & 0 \\ \vdots & & \ddots & & \vdots \\ w_{10,1} & 0 & \ldots & 0 & w_{10,11} \end{pmatrix} \tag{4.3}$$

Llamemos $O = O_1, O_2, \ldots, O_T$ a una secuencia de observaciones asociada a la señal que se usará para adaptar (la grabación real en este caso). Al igual que en el capítulo 2, llamemos λ al conjunto de parámetros asociados a la cadena que queremos adaptar. La probabilidad de la secuencia de observaciones bajo el modelo dado por el párametro λ, $\mathbb{P}(O|\lambda)$, pensada como función de λ, es la función objetivo a maximizar durante la adaptación. Si $\bar{\lambda}$ es el conjunto de parámetros con las medias adaptadas por la fórmula 4.2, en [40] se encuentra una expresión para

[13] Por ejemplo, la primera entrada del vector adaptado será, según las ecuaciones 4.2 y 4.3 $\hat{\mu_s}(1) = \omega w_{1,1} + w_{1,2}\mu_s(1)$. Si $\omega = 0$, la reestimación en ese caso será simplemente un múltiplo del valor original; en cambio, si $\omega = 1$, a ese múltiplo se le sumará la constante $w_{1,1}$. De ahí el uso del término *offset*.

la matriz de reestimación de parámetros que mejora esa probabilidad, es decir, $\mathbb{P}(O|\bar{\lambda}) > \mathbb{P}(O|\lambda)$.[14]

Para dar esa expresión de la matriz de reestimación de parámetros, es necesario definir algunas cantidades. Primero, se organizan los elementos no nulos de W_s en un vector v_s como sique:

$$v_s = \begin{pmatrix} w_{1,1} \\ \vdots \\ w_{10,1} \\ w_{1,2} \\ \vdots \\ w_{10,11} \end{pmatrix}$$

y se define la matriz D_s como la concatenación de dos matrices diagonales 10×10, que tienen en la diagonal a los elementos del vector de medias extendido ξ_s:

$$D_s = \begin{pmatrix} \omega & 0 & \dots & 0 & \mu_s(1) & 0 & \dots & 0 \\ 0 & \omega & \dots & 0 & 0 & \mu_s(2) & \dots & 0 \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \omega & 0 & 0 & \dots & \mu_s(10) \end{pmatrix}$$

Así, el vector de reestimación (que da las entradas de la matriz de transformación de las medias) se calcula como:

$$v_s = \left[\sum_{t=1}^{T} \gamma_t(s) D_s^t \Sigma_s^{-1} D_s\right]^{-1} \left[\sum_{t=1}^{T} \gamma_t(s) D_s^t \Sigma_s^{-1} O_t\right]$$

siendo $\gamma_t(s)$ la probabilidad de ocupar el estado s en tiempo t dada la secuencia de observaciones (definida por la ecuación 2.2 del capítulo 2, que ya vimos puede ser calculada a partir del algoritmo *forward-backward*) y Σ_s es la matriz de covarianza de la distribución de probabilidad de las observaciones del estado s.

Realizada la adaptación acústica, se vuelve a clasificar el audio real; los resultados aparecen en la tabla 4.5. Como allí se ve, el desempeño no mejora. Siguen clasificándose bien solamente dos compases, y la diferencia relativa sigue siendo muy baja en todos los casos.

Esto indica que la adaptación acústica que solo modifica las medias de las probabilidades de observación no es adecuada en este caso. Podría pensarse en realizar una adaptación más sofisticada: en lugar de una matriz de reestimación que tenga ceros en casi todas sus entradas (como indica la ecuación 4.3), se puede buscar una matriz cuyas entradas a priori no tengan por qué ser nulas. Esto asumiría que el vector de observaciones no tiene entradas independientes, hipótesis que puede llegar a ser más razonable en el caso real. También podrían, además de las medias, modificarse las varianzas, como se hace en [26]. Aparecen aquí posibles caminos por los cuales seguir investigando en trabajos futuros; se hablará de eso en el capítulo 5.

[14] Si se recuerda lo visto en el capítulo 2, se observará que este resultado no es más que un caso particular del algoritmo Baum-Welch, donde solo se reestiman las medias de las densidades de probabilidad de las observaciones.

Compás	*Ground truth*	*Clasificación (score)*	*Siguiente máximo (score)*	*Dif. relativa*
1	C	C (-6381.10)	R (-6708.93)	0.049
2	L	C (-6638.63)	R (-7845.71)	0.154
3	L	C (-6853.05)	R (-8392.90)	0.183
4	M	R (-3667.97)	C (-4349.25)	0.157
5	M	R (-4420.74)	C (-4551.78)	0.029
6	M	C (-4042.32)	R (-4099.48)	0.014
7	Desconocido	C (-8844.52)	M (-10573.28)	0.164
8	Desconocido	C (-9698.39)	R (-10246.05)	0.053
9	C	C (-6931.38)	R (-9337.90)	0.258
10	L	C (-6846.89)	R (-8659.23)	0.209
11	L	C (-7135.26)	R (-8788.98)	0.188
12	R	C (-7265.07)	R (-9106.04)	0.202
13	A	C (-6711.26)	R (-8007.05)	0.162
14	Desconocido	C (-7451.52)	R (-8899.38)	0.163
15	Desconocido	C (-7838.30)	R (-9028.89)	0.132
16	M	R (-4909.95)	C (-5041.07)	0.026
17	M	R (-3845.68)	C (-3954.19)	0.027
18	M	R (-3467.81)	C (-3509.57)	0.012
19	M	L (-1959.02)	R (-2834.55)	0.309

Tabla 4.5: Resultados de la clasificación de compases para el audio ARR1, luego de la adaptación acústica.

Clasificación de pulsos

Pasemos ahora a la clasificación de pulsos, nuevamente para el audio ARR1. Si se vuelve a su partitura (figura 4.19), se verá que hay varios pulsos similares a los de entrenamiento, ya sea con o sin adornos. Hay también pulsos que no se parecen en nada, por ejemplo, todos los correspondientes al vigésimoquinto compás (esos compases serán considerados "desconocidos", en línea con lo que se aclaró antes). Veamos qué sucede si las cadenas entrenadas con audios sintéticos se usan para clasificar los compases de esa pieza, evaluando el impacto de usar o no adaptación acústica.

La figura 4.21 muestra el histograma de las diferencias relativas entre los primeros dos máximos de la log-verosimilitud, para la clasificación sin usar adaptación acústica. En ese caso, el porcentaje de acierto en la clasificación fue de un 8.33 %. En los hechos, solo 4 de los 48 pulsos de la pieza se clasifican correctamente.[15] Al igual que para los compases, se obtiene un desempeño pobre, tanto en el porcentaje de acierto como en la confiabilidad de la clasificación.

Si se analizan los aciertos en la clasificación (los pulsos 1.1, 8.3, 13.1 y 14.1)

[15]La pieza tiene 19 compases, de los cuales 7 son de madera, por lo que para la clasificación de pulsos quedan 12 compases (o 48 pulsos).

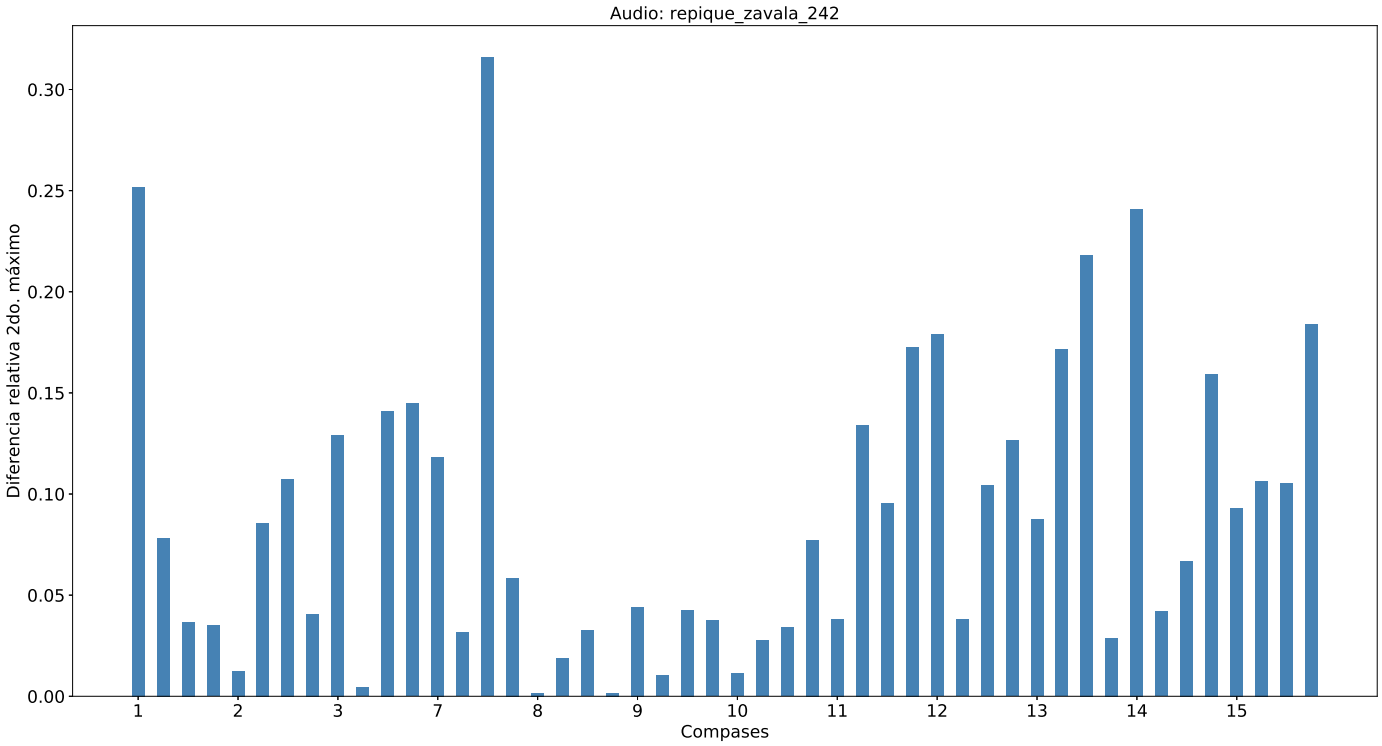


Figura 4.21: Diferencia relativa entre los primeros dos máximos de la log-verosimilitud, para la clasificación de pulsos del audio ARR1.

se constatará que en todos los casos se trata de pulsos de tipo *I*. Por ejemplo, la figura 4.22 muestra el visualizador para el primer pulso del decimocuarto compás. Allí se ve que, si bien es ese pulso es un *I* adornado (con el agregado de un golpe de palo), el sistema lo reconoce correctamente.

Sin embargo, otros pulsos que son como los de entrenamiento no se reconocen correctamente. Aún si miramos el porcentaje de acierto dentro de estos (es decir, sacamos de la ecuación a aquellos pulsos declarados "desconocidos") obtenemos un porcentaje bajo, del 11.11 % (4 en 36). A modo de ejemplo, se muestra en la figura 4.23 el visualizador para el segundo pulso del doceavo compás. Ese es un N_1 estándar, sin adornar; sin embargo, es reconocido como un *I*. Si se vuelve al histograma 4.21, se verá que en ese pulso la medida de confiabilidad es elevada (al menos comparada con la que se obtiene para el resto). Esta es tal vez la prueba más fehaciente del pobre desempeño del sistema al intentar clasificar un audio real.

Otro ejemplo claro de lo inadecuado del sistema para enfrentarse a audios reales es el pulso 7.3, que se muestra en la figura 4.24. Ese es un ejemplo de patrón desconocido; el sistema sin embargo lo clasifica como un *I*. Y no solo eso, si no que lo hace con la confiabilidad más alta para este audio. Por lo tanto, la medida de confiabilidad para este audio ni siquiera permite determinar si se trata de un pulso que no pertenece a ninguno de los patrones de entrenamiento.

Si se realiza adaptación acústica, los resultados no cambian significativamen-

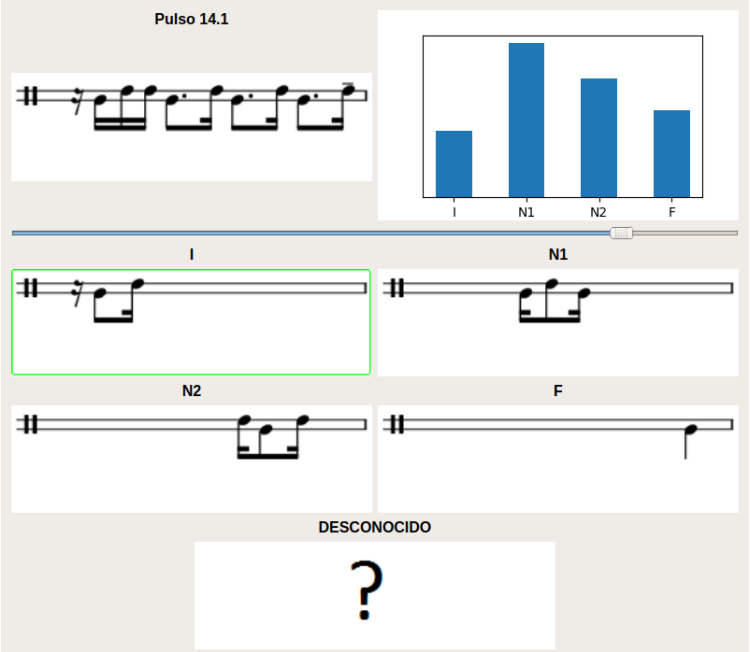


Figura 4.22: Visualizador de clasificación para el primer pulso del decimocuarto compás, para el audio ARR1.

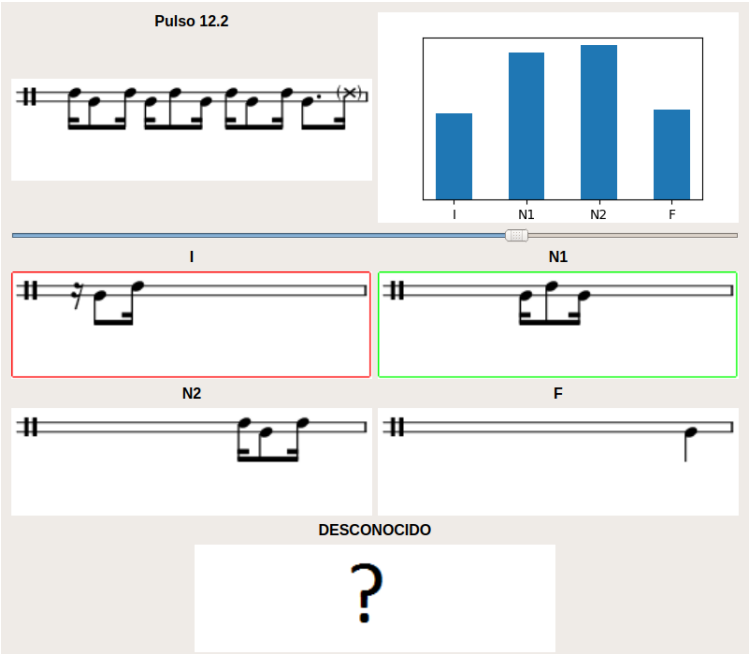


Figura 4.23: Visualizador de clasificación para el segundo pulso del doceavo compás, para el audio ARR1.

te. Se pasa a reconocer un pulso más correctamente (el 9.1, además de los que se reconocía antes), pero esencialmente la clasificación es la misma; la figura 4.25

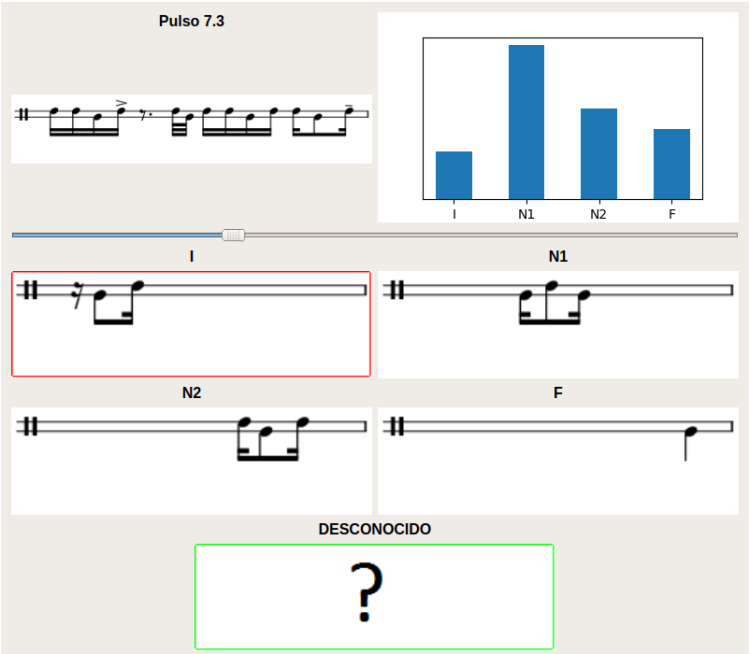


Figura 4.24: Visualizador de clasificación para el tercer pulso del séptimo compás, para el audio ARR1.

muestra las diferencias relativas. Lo único a destacar es que la confiabilidad en la clasificación aumenta, mostrando que la adaptación acústica realizada es perjudicial en este caso, y reforzando la idea de que se necesita una adaptación un poco más sofisticada. De todas formas, debe tenerse en cuenta lo que se está intentando lograr: se quiere que un sistema entrenado con audios sintéticos sea capaz de reconocer audios reales. Cualquier persona que sea consciente de la diferencia que se percibe entre ambos tipos de audios (de nuevo, se insta al lector a comparar auditivamente un audio sintético y uno real de los aquí discutidos), podrá comprender cabalmente la dificultad de esa tarea.

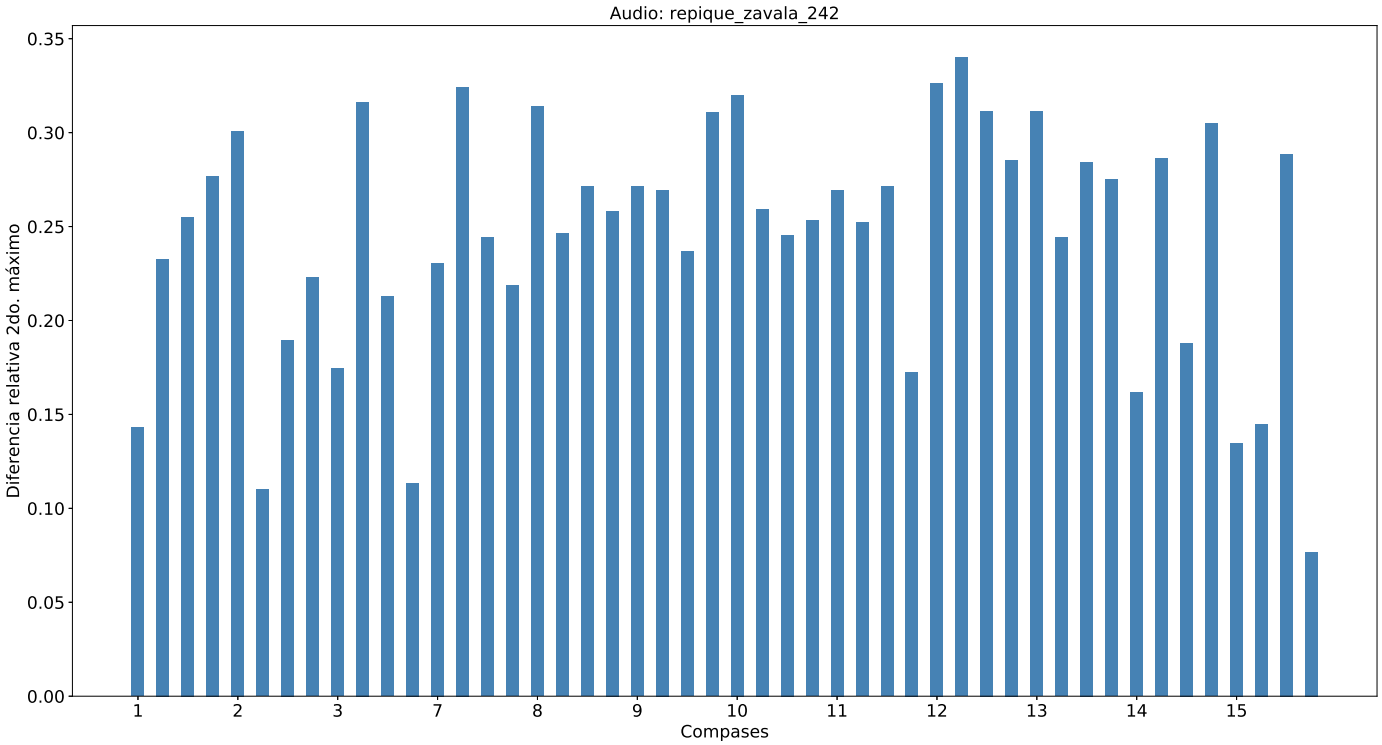


Figura 4.25: Diferencia relativa entre los primeros dos máximos de la log-verosimilitud, para la clasificación de pulsos del audio ARR1, con adaptación acústica.

4.2. Base de piano

En el caso del piano, recordemos que el problema planteado es reconocer si cada compás de una interpretación es o no una ejecución del patrón básico. Por lo tanto, aquí la clasificación se hace siempre a nivel de compases.

Dado el esquema de clasificación planteado en la sección 3.2.2, se va a entrenar una sola HMM para que reconozca la base. Así, para entrenar esa cadena se necesita tener varias realizaciones de la base, es decir, es necesario contar con audios cuyos compases de base se encuentren identificados. Este requerimiento es bastante más laxo que para el caso del repique, ya que para un oído apenas entrenado es fácil separar qué es base de lo que no lo es, mientras que la identificación de patrones de repique requiere de conocimiento musicológico (pues es necesario transcribir la interpretación e identificar los patrones en la partitura). Por lo tanto, generar una base de datos a partir de audios reales es más simple para este problema que para el reconocimiento de variantes de repique: si para esos audios se cuenta con los tiempos de comienzo de cada compás, basta escucharlos y determinar si cada uno es o no base.

Siguiendo esa idea, se etiquetaron 5 audios de interpretaciones reales.[16] Todas las grabaciones corresponden a un mismo intérprete de piano, Gustavo Oviedo, músico de candombe identificado con el estilo del barrio Palermo, y uno de los tocadores de piano más influyentes en las generaciones posteriores.[17] Los datos de los audios se resumen en la tabla 4.6.

Allí se muestra, para cada archivo, la duración en minutos y segundos y la cantidad de intérpretes en la toma. Se puede observar que hay tomas con 3 o con 4 intérpretes. Estas tomas se corresponden a dos configuraciones de tambores: en el caso de 3 intérpretes, la configuración es piano, repique y chico; en el caso de 4, se agrega un repique. Si bien se cuenta con audios separados por canales (uno por cada tambor), en las tomas de 4 el ruido cruzado es sensiblemente mayor que para las de 3, ya que en el primer caso los intérpretes se encontraban más cerca entre sí. Por lo tanto, usar los audios con 4 tambores puede llegar a afectar negativamente el desempeño del sistema de clasificación. Como contraparte, usarlos permite tener más instancias con las que clasificar, y con más variabilidad en sus condiciones de grabación, lo que puede derivar en un sistema que permita clasificar audios grabados en diferentes condiciones. En las pruebas que se realizaron se tuvo en cuenta esta dicotomía, evaluando cómo funciona el reconocimiento si se usan o no estas tomas.

En la tabla 4.6 también se muestra la cantidad de compases según el etiquetado. La división se hizo en cuatro clases: “Base”, “Variación” de la base, “Madera” y “Otros”. Esta última categoría se creó pues hay tomas en las que el intérprete

[16] Esos audios fueron registrados en la misma sesión de grabación que el audio real usado en el reconocimiento de variantes de repique; ver [61] por más información.

[17] Gustavo, junto a su hermano Edinson “Palo” Oviedo y Fernando “Hurón” Silva participaron de varios discos de Jaime Roos en los 80. Algunos de los temas más notorios: “Tal vez Cheché” y “Pirucho”, del disco “Mediocampo” (1984), “El tambor”, del disco “7 y 3” (1986), y “Candombe de Reyes” del disco “Sur” (1987). En todos ellos Gustavo tocó el piano.

			Cantidad de compases				
Archivo	*Duración*	*Intérpretes*	*Base*	*Variación*	*Madera*	*Otros*	*Total*
Take_211	3:29.43	3	68	10	5	1	84
Take_212	2:52.52	3	63	14	5	2	84
Take_213	3:04.19	3	72	18	5	0	95
Take_221	2:53.12	3	73	16	2	2	93
Take_222	1:55.01	3	46	12	2	1	61
Take_311	3:48.45	4	92	21	4	2	119
Take_312	3:01.73	4	76	16	4	2	98
	Total	490	107	27	10	634	

Tabla 4.6: Información de los audios usados para el reconocimiento de base de piano.

ejecuta patrones que no se pueden considerar como base, y tampoco como una variación de la misma. En la práctica, esos compases se corresponden o con el compás final de la pieza (en el que el intérprete ejecuta una figura que podríamos llamar de finalización, que pretende darle un cierre a la ejecución) o con un compás que se ejecuta entre la madera y la base, cuya finalidad es "enganchar" la subida del toque de madera al toque en la lonja.

En esta parte se pretenden identificar los compases de las columnas 3 y 4, es decir, los compases que sean el patrón típico de piano o una variación. Así, el resto de los compases no se utilizan en ningún momento, ni en el entrenamiento ni en la clasificación.

La evaluación de desempeño se realizará mediante el método de validación cruzada, en modalidad *leave-one-out*: la HMM que reconocerá la base se entrena con todos los archivos menos uno,[18] y se evalúa el desempeño sobre el restante (que en este contexto es referido usualmente como archivo de *test*). Los conjuntos de entrenamiento y evaluación de desempeño se van rotando de manera de cubrir todas las combinaciones posibles.

Aquí vale la pena recordar que el entrenamiento se lleva a cabo usando solo los compases etiquetados como "Base" del conjunto de entrenamiento, ya que es para reconocer ese patrón que se entrena la cadena. La evaluación de desempeño se lleva a cabo sobre todos los compases etiquetados como "Base" o "Variación" del archivo de test. Por ejemplo, si se incluyen las tomas de 4 tambores en el entrenamiento y se fija el archivo Take_211 como test, el conjunto de entrenamiento estará formado por 422 compases (los 490 compases de base en el conjunto total menos los 68 compases de base del archivo de test), y se evaluará el desempeño del sistema sobre los 78 compases del archivo de test etiquetados como base o variación (68

[18]Cuántos archivos se utilicen para entrenar dependerá de si se usan o no las tomas con 4 tambores.

de base y 10 de variaciones).

Hechas estas puntualizaciones, los resultados de la clasificación se presentan a continuación. Las pruebas realizadas se dividen en dos grupos: aquellas en las que las tomas de 4 tambores no se usan, y aquellas en las que sí se usan. A su vez, dentro de cada grupo se analizó el desempeño del sistema usando dos frecuencias máximas para el cálculo de las características (*spectral flux* y MFCCs) de la señal: 2 y 4 kHz. Dado que el piano es el tambor de sonido más grave, se ubicará en el rango más bajo de frecuencias. Por lo tanto, es de esperar que la frecuencia máxima de análisis en el procesamiento de tiempo corto juegue un papel fundamental en el reconocimiento: cuanto más alta sea, más información de los otros tambores se estará analizando en el espectro, por lo que, en las tomas donde el ruido cruzado sea elevado (como es el caso de las de 4 tambores), una frecuencia de análisis un poco menor debería permitir un mejor desempeño. Aquí hay un compromiso que debe hacerse, ya que disminuir la frecuencia máxima implica analizar una menor porción del espectro, y por lo tanto disminuye la información que puede extraerse.

En la Tabla 4.7 se muestran los resultados obtenidos para ambas frecuencias máximas de análisis, según si se usan o no las tomas de 4 tambores. La primera columna indica el archivo con el cual se evalúa desempeño. En el caso de usar las tomas de 4, la cadena es entrenada con los compases de base de las 5 tomas restantes; en el caso que no se utilicen, es entrenada con las 4 tomas de 3 tambores que no sean el archivo de test. Como medida de desempeño se usa la f-measure, que ya fue introducida en la sección 3.2.2.

	f-measure			
	Con tomas de 4		*Sin tomas de 4*	
Archivo de test	$f_{max} = 2$ *kHz*	$f_{max} = 4$ *kHz*	$f_{max} = 2$ *kHz*	$f_{max} = 4$ *kHz*
Take_211	0.962	1.000	0.955	0.962
Take_212	0.992	0.967	0.984	0.992
Take_213	0.980	0.986	0.980	0.986
Take_221	0.993	0.993	1.000	1.000
Take_222	0.958	0.958	0.979	0.939
Take_311	0.963	0.953	–	–
Take_312	0.897	0.877	–	–

Tabla 4.7: f-measure para la clasificación de la base de piano, según distintas frecuencias máximas de análisis.

Respecto a los resultados obtenidos, en primera instancia podemos observar que el desempeño es bueno para todos los casos: en general, se obtiene una f-measure cercana a 1 (y en todos los casos es mayor a 0.87). Si las tomas de 4 no se utilizan, los resultados no cambian sustancialmente con las distintas frecuencias

máximas de análisis: el desempeño con 4 kHz mejora para casi todas las tomas respecto al obtenido para 2 kHz, pero se mantiene en valores muy similares. Si las tomas de 4 son utilizadas, el desempeño con 2 kHz es apenas superior. Esto queda claro sobre todo en el caso de los audios *Take_311* y *Take_312*, que son las que tienen 4 intérpretes: el f-measure es más alto respecto al obtenido para 4 kHz (aunque por muy poco). Eso es razonable, pues bajar la frecuencia máxima disminuye la incidencia del ruido cruzado.

Si bien el desempeño al usar tomas de 4 tambores disminuye cuando se clasifican tomas de 3, la disminución no es significativa, y como contrapartida, usar tomas de 4 permite tener más instancias para el entrenamiento, y un sistema que puede funcionar en casos más generales que solo para tomas con 3 tambores. Así, es razonable usar las tomas de 4, y mantener una frecuencia máxima de 2 kHz.

En la tabla 4.8 se analizan con más detalle los resultados de la clasificación para ese caso (frecuencia máxima de 2 kHz, usando las tomas de 4 tambores). Allí se reportan distintas métricas usuales en problemas de clasificación binaria: además del *precision*, el *recall* y la f-measure introducidas en la sección 3.2.2, se reporta el *accuracy* en la clasificación, definido como la proporción de las instancias que son correctamente clasificadas. Si tp es la cantidad de instancias correctamente clasificadas como "Base" (*true positives*), tn es la cantidad de compases correctamente clasificados como "Variación" (*true negatives*), y M es la cantidad total de compases, el accuracy se calcula:

$$acc = \frac{tp + tn}{M}$$

Archivo de test	*Accuracy*	*Precision*	*Recall*	*f-measure*
Take_211	0.936	1.000	0.926	0.962
Take_212	0.987	0.984	1.000	0.992
Take_213	0.967	0.960	1.000	0.980
Take_221	0.989	1.000	0.986	0.993
Take_222	0.931	0.920	1.000	0.958
Take_311	0.938	0.938	0.989	0.963
Take_312	0.826	0.875	0.921	0.897

Tabla 4.8: Resultados de la clasificación de la base de piano usando tomas de 4 tambores, para una frecuencia máxima de 2 kHz.

Así, los resultados de la tabla 4.8 muestran que en general el porcentaje de acierto en la clasificación es alto: el peor desempeño se obtiene cuando se usa el *Take_312* como archivo de test: allí el *accuracy* es 82.6 %. Que se obtenga el peor desempeño para este archivo es razonable, ya que si se escucha la grabación, se observará que este audio es el que tiene un mayor nivel de ruido proveniente de los otros tambores (especialmente del chico).

Sin embargo, a pesar de la baja *accuracy*, vemos que para ese audio el *precision* y el *recall* resultan cercanos al 90 %. Recordemos que el *precision* es la proporción de compases correctamente clasificados como "Base" respecto al total de los compases clasificados así; que sea de 0.875 para ese audio indica que el 87.5 % de los compases clasificados como "Base" efectivamente lo son, pero no dice nada respecto a cuántos compases de base fueron mal clasificados como variaciones. Esa información de alguna manera la mide el *recall*, que se define como la proporción entre las instancias correctamente clasificadas como "Base" y el total de instancias de la clase "Base". Si en este caso es de 0.921, esto quiere decir que el 92.1 % de los compases de base son bien clasificados.

Para el resto de los archivos, el desempeño es muy bueno, obteniendo un *f-measure* superior a 0.95 en todos los casos. Destacables son los resultados obtenidos cuando se usa el *Take_311* como test. Recordemos que este audio se corresponde con una grabación con 4 tambores. Que obtenga tan buenos resultados con un sistema entrenado mayoritariamente con audios de 3 tambores (salvo el *Take_312*, el resto de los audios de entrenamiento lo son) es sumamente alentador respecto a las posibilidades del sistema de clasificar audios cuyas condiciones de grabación sean diferentes a las de los audios usados para entrenarlo.

Para poder analizar de mejor forma los resultados de la clasificación, se generó un programa que permite visualizarlos; la Figura 4.26 muestra una captura de este visualizador. La primera fila muestra un mapa que indica el valor del flujo espectral normalizado para cada *tactus* de la pieza. Este valor se muestra en escalas de grises, donde el negro representa 1 y blanco 0. Así, cuanto más oscuro sea el rectángulo asociado a un *tactus*, más probable es que haya un golpe en ese lugar. En el eje horizontal se suceden los compases, y en el vertical los *tactus* de cada compás. Esta representación es usual en análisis de patrones rítmicos de piano, como por ejemplo en [58]. En esa primera fila se indica además, con distintos colores, el *ground-truth* de la pieza, es decir, a qué clase pertenece cada compás ("Base" o "Variación"). También, se muestra gráficamente el resultado de la clasificación: aquellos compases mal clasificados se marcan en este diagrama con un rectángulo con rayas diagonales negras.

En la segunda fila de 4.26 se muestra el histograma acumulativo y normalizado de las verosimilitudes obtenidas a la salida de la HMM, calculado para todos los compases de la pieza a clasificar (los etiquetados como "Base" o como "Variación"). En ese mismo gráfico se marca el umbral de clasificación, calculado según lo discutido en la sección 3.2.2. Recordemos que se clasificarán como base todos los compases cuya verosimilitud caiga en un *bin* del histograma que supere este valor; los que no lo hagan se clasificarán como variaciones.

El visualizador permite seleccionar un compás en el mapa de la primera fila. Al hacerlo, es posible escuchar el audio correspondiente a ese compás (lo que se logra apretando el botón "play" o apretando la barra espaciadora). Además, una vez seleccionado, se resalta en el histograma el *bin* donde se ubica la verosimilitud obtenida para ese compás. En la captura de la figura 4.26, se seleccionó el compás 32, que es una variación mal clasificada como base. Allí vemos que la verosimilitud para este compás se ubica en un nivel muy cercano (pero superior) al umbral, por

lo que el error de clasificación es razonable en ese caso. Si se repite esta exploración para el resto de los compases mal clasificados, se verá que siempre la verosimilitud en esos casos cae muy cerca del umbral. Esto, mirado de otra perspectiva, quiere decir que la mayoría de los compases son bien clasificados, y obtienen una verosimilitud lejana a la del umbral (muy por encima si son "Base", muy por debajo si son "Variación"). En el Apéndice A se explica cómo utilizar este visualizador para que el lector pueda comprobar por sí mismo las observaciones aquí hechas.

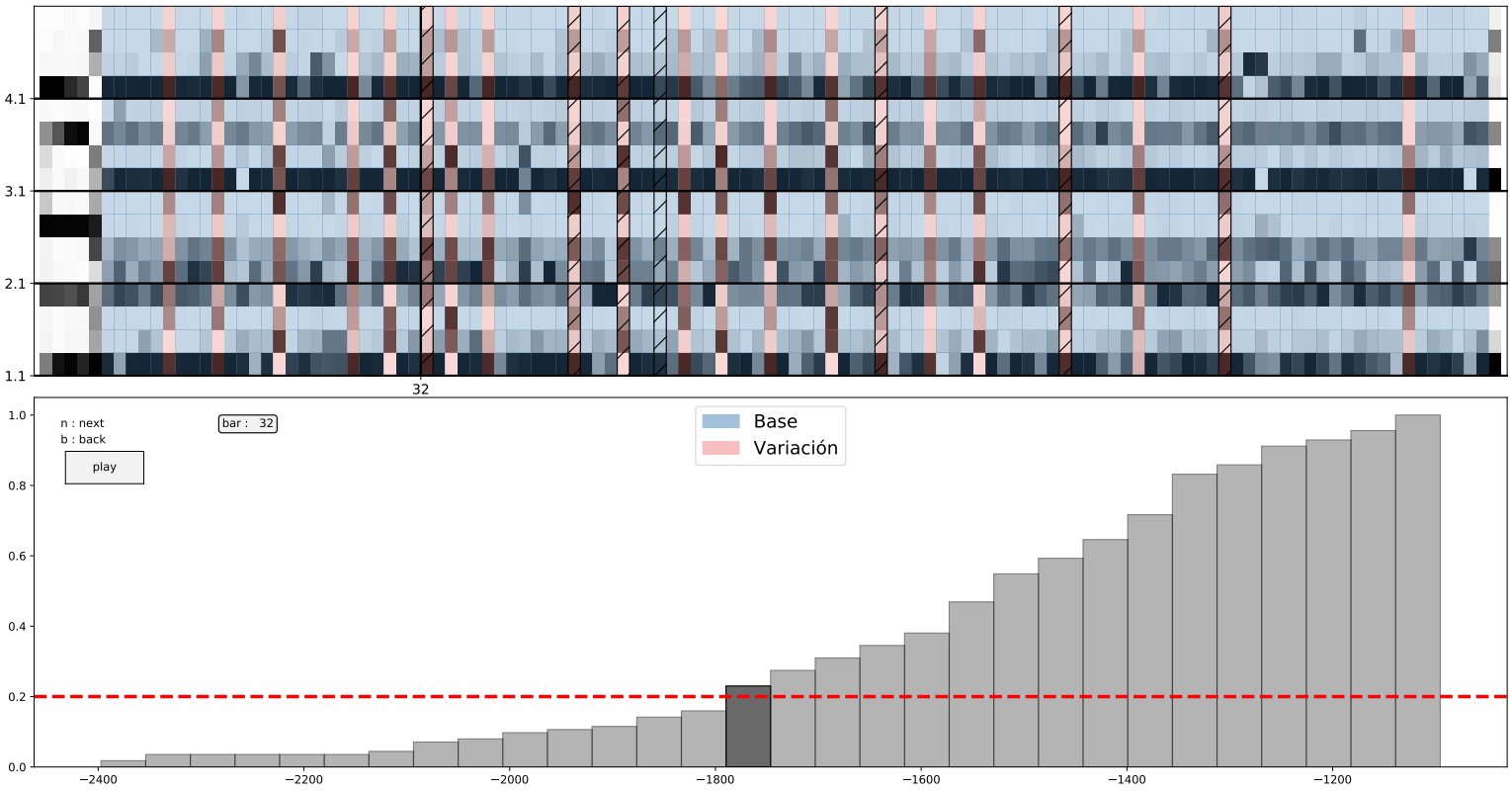


Figura 4.26: Visualizador para la clasificación de base de piano, para el audio *Take_312*.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Discusión, conclusiones y trabajo futuro

En este trabajo de tesis se presentó una propuesta de metodología para el reconocimiento automático de patrones rítmicos a partir de señales de audio. Los experimentos reportados se concentran en el ritmo del candombe, específicamente para los tambores repique y piano. La propuesta se centra en la utilización de cadenas ocultas de Markov como herramienta de clasificación, utilizando características derivadas del audio (mas específicamente, del espectro de la señal de audio) como observaciones de esa cadena. Los estados ocultos de la cadena, y las transiciones permitidas entre estos, se modelan a partir de los patrones rítmicos que se quieren reconocer, teniendo en cuenta que esos patrones son secuencias de golpes de palo y mano en el tambor: los distintos estados ocultos están asociados a los distintos golpes, y las transiciones permitidas entre estados tratan de modelar cómo se suceden esos golpes dentro de cada patrón rítmico.

Además de exponer la metodología elegida, explicando cada etapa y fundamentando las decisiones tomadas, se expusieron los resultados experimentales que se obtienen de aplicar esa metodología. En cada parte de ese proceso, han surgido posibles caminos a recorrer a futuro, y se plantearon algunas interrogantes respecto al enfoque elegido (y a posibles enfoques alternativos). En este capítulo se pretende profundizar un poco más en la discusión de estos puntos.

En primer lugar, los resultados obtenidos para la clasificación de patrones de repique permiten visualizar el alcance del enfoque elegido: para el entrenamiento y la clasificación con audio sintético, el porcentaje de acierto en la clasificación es muy alto, indicando que la metodología es adecuada para este problema, al menos en el caso de que los audios a clasificar tengan características acústicas similares a los audios con los que las cadenas fueron entrenadas. Ya hemos visto cómo decae el desempeño en el caso de los audios reales. Surge entonces como posible continuación natural trabajar para lograr un sistema que pueda ser entrenado con audios reales. Esto implica generar una base de datos anotada, es decir, un conjunto de grabaciones de repique que cuenten con información de tiempo de comienzo y fin de cada pulso, ubicación temporal de los golpes, y una transcripción, asignando cada pulso (o compás) a algunas de las clases que se quieren reconocer.

Otra alternativa en ese punto es mejorar el proceso de adaptación acústica para ajustar las cadenas entrenadas con audio sintético. El camino explorado en

ese sentido consistió en modificar solamente las medias de las densidades de observación de cada HMM, y un paso natural para complejizar ese proceso es pensar en modificar también las varianzas de esas densidades, como fue mencionado en el capítulo 4. Un paso intermedio sería modificar solamente las medias, pero de una manera más sofisticada que la utilizada aquí: recordemos que en la adaptación se buscaba una reestimación de la forma $\hat{\mu_s} = W_s \xi_s$, donde la matriz de ajuste utilizada era

$$W_s = \begin{pmatrix} w_{1,1} & w_{1,2} & 0 & \dots & 0 \\ w_{2,1} & 0 & w_{2,3} & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ w_{10,1} & 0 & \dots & 0 & w_{10,11} \end{pmatrix}$$

y $\xi_s = [\omega, \mu_s(1), \mu_s(2), \dots, \mu_s(10)]^t$. Así, cada media reestimada consiste en un múltiplo de la media correspondiente más una constante (un *offset* dado por la variable binaria ω). Podría pensarse en buscar una matriz completa, es decir, una matriz que a priori no tenga casi todas sus entradas nulas, haciendo que la reestimación sea una combinación lineal de todas las entradas del vector original (método conocido como *full matrix mean transformation*). La principal desventaja que tiene esa idea es el aumento del costo computacional, además de que, como se muestra en [40], esto deriva en ecuaciones implícitas para la matriz de reestimación de parámetros.

De todas maneras, los resultados obtenidos para el tambor piano, donde se usó una metodología muy similar, muestran que entrenar las cadenas con audios reales lleva a buenos resultados en la clasificación. Así, si bien la idea de generalizar un sistema entrenado con audio sintético es muy seductora, en el corto plazo probablemente sea más sensato centrarse en la generación de datos anotados. En ese sentido, en 2018 se publicó un trabajo en el que se expone una base de datos de samba [44] que cuenta con algunos datos que cumplen la mayoría de los requerimientos necesarios para aplicar la metodología aquí propuesta (archivos de audio con un instrumento por canal, información de tiempo de comienzo y fin de cada pulso, ubicación temporal de los golpes). Dado que ese estilo musical utiliza varios instrumentos de percusión, este enfoque puede ser adapatado para el reconocimiento se patrones rítimicos de samba. Por la cercanía personal que hay con algunos de los autores de ese artículo ya se han hecho contactos en ese sentido, y se espera que eso pueda derivar en trabajos conjuntos en un futuro cercano.

Respecto a la metodología, existen varias alternativas posibles a lo hecho, que van desde complejizar algunos procesos hasta la utilización de enfoques totalmente distintos para algunas tareas. Dentro del primer grupo, por ejemplo podría pensarse en revisar la topología impuesta a las transiciones entre estados ocultos. Recordemos que se utiliza una cadena de izquierda a derecha, donde cada estado oculto está asociado a un silencio o a un golpe. Tal vez esa topología sea muy restrictiva, pues implica que dentro de cada pulso o compás existe una cantidad fija de golpes, y ya vimos que algunos adornos implican aumentar la cantidad de golpes. Eso causaba que, al intentar clasificarlos, la cadena asimile varios golpes a un mismo estado oculto, para poder mantener el recorrido de los estados ocultos de izquierda a derecha. Además, en la topología elegida, siempre la sucesión de

estados ocultos estaba forzada a ser "golpe-silencio-golpe" etc..., lo que no permite la transición directa entre dos estados asociados a golpes distintos, y se deba en cambio pasar siempre por un silencio en el medio. Tal vez una posible solución a eso sea mantener la estructura de izquierda a derecha, pero permitiendo transiciones entre el estado actual y uno posterior cualquiera (recordemos que las únicas entradas no nulas de la matriz de transición de estados eran la diagonal, y la diagonal superior, por lo que las únicas transiciones posibles son mantenerse en el mismo estado o moverse al estado inmediatamente posterior). Eso implicaría buscar una matriz de transición de estados triangular superior, aumentando significativamente la cantidad de parámetros a estimar. Una solución intermedia puede ser definir una "longitud máxima de salto" en la cadena oculta, es decir, imponer que las transiciones permitidas sean entre el estado actual y, por ejemplo, los tres estados posteriores. Así, las únicas entradas no nulas de la matriz de transición serían aquellas de la forma $a_{i,i+n}$, para $n = 0, 1, \ldots, N$, siendo N la longitud máxima de salto permitida (en el ejemplo de tres estados, sería $N = 3$).

Otra alternativa en ese sentido, que implicaría un cambio de enfoque, sería tomar una topología como la impuesta en [55]. En ese trabajo, que usa HMMs para ubicación y reconocimiento de golpes de batería, se entrena una cadena oculta por cada tipo de golpe, más una cadena para el silencio (esas cadenas son entrenadas independientemente, es decir, la cadena asociada a un tipo de golpe se entrena independientemente de las asociadas a los otros golpes o al silencio). Luego, esas cadenas pueden ser combinadas arbitrariamente, como se muestra en la figura 5.1. Así, existe una suerte de jerarquía en el proceso de entrenamiento: primero se entrenan las cadenas para cada golpe y para cada silencio, y luego se entrena el modelo que dicta cómo pueden combinarse esas cadenas.

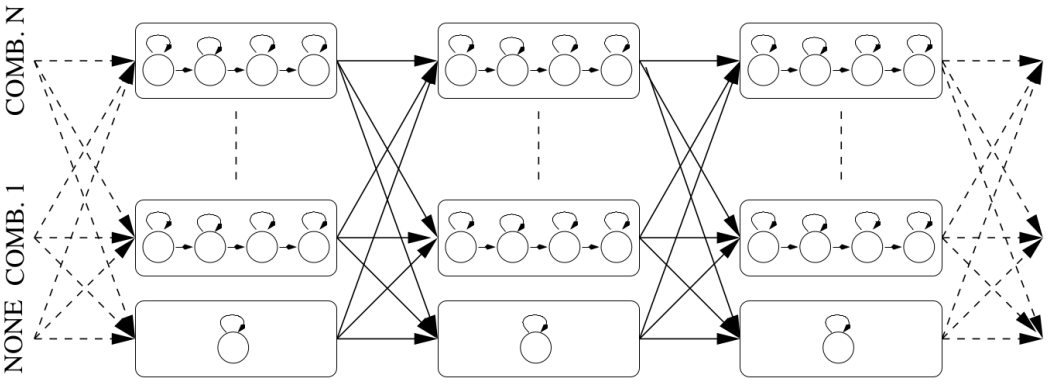


Figura 5.1: Topología de HMMs utilizada en [55]. Figura extraída de ese trabajo. COMB. i representa la cadena entrenada para el tipo de golpe i, mientras que NONE sería la que representa al silencio.

Aplicada a este trabajo, esa metodología podría funcionar de la siguiente manera: primero se segmentarían los audios de entrenamiento en tres clases: golpes de mano, golpes de palo, y silencio. Cada segmento se utilizaría para entrenar las

cadenas correspondientes: por ejemplo, todos los segmentos identificados como golpes de palo se usarían para entrenar una cadena asociada al golpe de palo, etc. Una vez entrenadas esas tres cadenas, se determinarían las posibles transiciones entre ellas, identificando en los audios cómo se suceden los golpes de palo, los golpes de mano y los silencios, o bien podrían permitirse transiciones arbitrarias entre ellas.

También, y como se vio en la sección 1.5 esto es lo que se hace en otros trabajos que usan HMMs para reconocimiento de instrumentos de percusión, podría realizarse una decodificación similar al algoritmo de Viterbi. La idea es la siguiente: cada segmento de audio se evalúa según las tres cadenas (mano, palo o silencio). Así, para cada instante se tiene una probabilidad de estar en un golpe de palo, en uno de mano, o en un silencio. La idea de utilizar Viterbi es buscar el "camino óptimo" dentro de esa secuencia de probabilidades.[1] El problema con esto es que al final de esa etapa lo que se tiene es una sucesión de golpes o silencios, pero no aparece explícitamente la información sobre la ubicación de los patrones rítmicos, que es el objetivo que aquí se persigue. Determinar qué patrones rítmicos se ejecutaron implicaría entonces una etapa posterior de procesamiento, en la que esa secuencia de golpes y silencios sea traducida a una secuencia de patrones rítmicos, el equivalente a una etapa de análisis sintáctico en los problemas de voz hablada, complejizando aún más el proceso.

Si se opta por mantener el enfoque, y no realizar esta decodificación "estilo Viterbi", de todas maneras podría imponerse una etapa posterior de análisis sintáctico, en la que se analice si la sucesión de patrones rítmicos es musicalmente coherente. Incluso esa etapa podría incluirse dentro del proceso de clasificación, en un proceso iterativo: una primera sucesión de patrones rítmicos es analizada sintácticamente y, en caso de no respetarse alguna regla (como que después de un N_1 haya un I, que según las reglas de Jure no puede suceder en el caso de reconocimiento de pulsos) la etapa de reconocimiento sea repetida teniendo esa información. Nuevamente, eso complejiza el proceso, pero dado que la alternativa explorada aquí para imponer coherencia en el reconocimiento (reconocer compases en lugar de pulsos) no mejora sustancialmente la clasificación, una mayor complejidad parece ser necesaria.

A más bajo nivel, existen otras alternativas que pueden probarse, alejándose en menor o mayor medida del enfoque elegido. La que tal vez implique menos alejamiento es modificar las distribuciones de las observaciones de cada HMM. En la sección 3.1.5 se estableció que la distribución asumida fue una única normal en $\mathbb{R}^{10}$, y se analizó la pertinencia de esa asunción, concluyendo que, si bien no es descabellada, tal vez no sea la más adecuada. Un paso natural sería entonces considerar distribuciones que sean una mezcla de gaussianas. El inconveniente en este caso, además del aumento de parámetros a estimar en el entrenamiento, es que el toolbox `hmmlearn`, que es la implementación en Python de HMMs que se usó para este trabajo, tiene problemas de implementación que causan que los métodos iterativos para la reestimación de parámetros no converjan. Un análisis de las implementaciones disponibles de estos algoritmos muestra que no existen

[1] La sección VI. Connected Word Recognition Using HMMs del tutorial de Rabiner [56] explica en detalle cómo realizar esa decodificación.

otras alternativas libres y de código abierto que permitan mezclas de gaussianas u otras distribuciones. Existe por lo tanto una oportunidad de contribuir en ese aspecto, mejorando las implementaciones existentes de esos algoritmos, y es una tarea a explorar a futuro.

La siguiente alternativa que implica complejizar un poco más el enfoque elegido es la de modificar la distribución de probabilidad de la duración de estados ocultos de las HMM, lo que se conoce como cadenas de Markov de duración explícita. En la formulación usual de HMMs, si un estado i tiene asociada una probabilidad a_{ii} de mantenerse en ese estado, entonces la probabilidad de que se mantenga en ese estado durante d observaciones consecutivas es $p_i(d) = a_{ii}^d(1 - a_{ii})$. Las HMMs de duración explícita pretenden modificar el marco conceptual de las HMMs para admitir cualquier distribución $p_i(d)$. Introducir cadenas de duración explícita implica modificar los procedimientos de reeestimación de parámetros en el entrenamiento, aumentando la cantidad de parámetros a estimar. Además, dado que no existen implementaciones libres y de código abierto de este tipo de HMMs, es otra oportunidad para realizar contribuciones a la comunidad.

Asociado a eso, otro paso que implica un nivel de complejidad aún mayor es utilizar cadenas de Markov de mayor orden, es decir, cadenas donde la probabilidad de transición entre estados no dependa solamente de los estados en tiempo t y $t+1$, si no que lo haga de todos los estados anteriores, hasta tiempo $t-M$ (donde M está fijo y es el parámetro que define el orden de la cadena). Si bien es un paso natural a plantearse, vale la pena antes explorar las otras alternativas mencionadas, pues esta realmente implicaría una modificación importante del marco conceptual sobre el que se construyó todo el trabajo.

Finalmente, volviendo un poco a las modificaciones o cambios de enfoque de más alto nivel, un posible camino a seguir es modificar la clasificación de patrones de piano. En particular, la asignación de cada compás a las clases "Base" o "Variación" se hace mediante la umbralización del histograma de las log-verosimilitudes obtenidas, según lo explicado en la sección 3.2.2. Una alternativa a esto sería que la separación en esas clases se realice con algún método de agrupamiento no supervisado (es decir, pensar esa separación como un problema de *clustering*), como por ejemplo *k-means*.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Código adjunto y datos

En este apéndice se explica más detalladamente los pasos necesarios para ejecutar el código escrito para la tesis, incluyendo los pasos para utilizar los distintos visualizadores de resultados que se expusieron en el capítulo 4. Además, se incluyen los archivos de audio utilizados, para que el lector pueda escucharlos si así lo desea.

Por cualquier problema, duda o comentario, se puede contactar al autor en bmarenco@fing.edu.uy.

A.1. Prerrequisitos para utilizar el código

El código adjunto, que se puede descargar en el siguiente link, está escrito en Python 2.7, por lo que se necesita tener instalado un intérprete acorde para utilizarlo. Además de algunos paquetes usuales (NumPy, SciPy, Matplotlib), se usa la biblioteca `librosa` [10] para el cálculo de los MFCCs, y la biblioteca `ra` [49] para el cálculo del *spectral flux*. Para todo lo referente a HMMs, se utiliza la biblioteca `hmmlearn`. A continuación se listan las dependencias necesarias para usar el código, y se incluyen los comandos que deben ejecutarse en un terminal de Ubuntu 18.04 para instalarlas:

- Build essential: para compilar cualquier paquete en una distribución basada en Debian, es necesario contar con un compilador GCC/g++. Esas utilidades se encuentran todas en el paquete `build-essential` en Ubuntu, y en la versión 18.04 se instalan ejecutando en un terminal:[1]

  ```
  apt install build-essential
  ```

- Git: para instalar la última versión de algunos paquetes de la paǵina de los desarrolladores, es necesario tener instalado el programa de control de versiones Git. Para ello, el comando es:

  ```
  apt install git
  ```

[1] Probablemente estos comandos deban ser ejecutados como *super user*, es decir, incluyendo el término `sudo` antes del comando.

- Python 2.7: dado que el código está escrito en Python, es necesario contar con un intérprete acorde, y tener además el paquete de desarrollador. Los comandos para instalar ambos son:

```
apt install python
apt install python-dev
```

- Pip, NumPy y SciPy: el procesamiento utiliza herramientas de los paquetes NumPy y SciPy para algunas operaciones numéricas. Además, el paquete Pip permite instalar más fácilmente paquetes de Python. Para instalarlos basta ejecutar:

```
apt install python-pip
apt install python-numpy
apt install python-scipy
```

- Otros paquetes: se deben instalar otros paquetes de Python para que el código pueda ser usado. A continuación se listan esos paquetes y cómo instalarlos:
  - Scikit-learn: `pip install scikit-learn`
  - Pandas: `pip install -U pandas`
  - Libsndfile: `apt install libsndfile1 && apt install libsndfile1-dev`
  - Scikits.audiolab: `pip install -U scikits.audiolab`
  - Tkinter: `apt install python-tk`
  - PyQt4: `apt install python-qt4`
  - Matplotlib: `pip install matplotlib`
  - Tables: `pip install tables`
  - PyGames: `pip install pygames`

- Librosa: la librería `librosa` es la que se utiliza para el cálculo de los MFCCs. Se puede instalar a través de `pip`,[2] aunque se recomienda bajar la última versión disponible en el git del paquete y compilarla manualmente.[3] Para ello es necesario clonar este repositorio, ejecutando:

```
git clone https://github.com/librosa/librosa.git
```

  Si ese comando se ejecuta tal cual está escrito, se creará un directorio `librosa` en el lugar donde se haya ejecutado. Si se prefiere otra ubicación, basta con indicarla luego del comando:

```
git clone https://github.com/librosa/librosa.git ubicacion
```

[2] Ver la documentación disponible en `https://librosa.github.io/librosa/install.html` por más información.

[3] La url del github de librosa es `https://github.com/librosa/librosa`.

donde `ubicacion` es el *path* a donde se quieren clonar los archivos del paquete. Una vez clonado el contenido del git, se debe ir al directorio donde se extrajeron los archivos (i.e. `cd librosa/` o `cd ubicacion/`) y ejecutar:

```
python setup.py build
python setup.py install
```

El primer comando compila el paquete, mientras que el segundo lo instala.

- Hmmlearn: el paquete `hmmlearn` es el utilzado para todo lo referido a HMMs. Se recomienda compilarlo manualmente, de forma similar a `librosa`. Primero, clonar el repositorio git haciendo:

  ```
  git clone https://github.com/hmmlearn/hmmlearn.git
  ```

  luego ir al directorio donde se clonaron los archivos (`cd hmmlearn/`) y compilarlo e instalando, diciendo:

  ```
  python setup.py build
  python setup.py install
  ```

- Ra: es el paquete utilizaco para el cálculo del *spectral flux*. Para instalarlo, primero descargarlo en un `.zip` de la siguiente url:

  ```
  www.fing.edu.uy/~bmarenco/maestria/downloads/ra.zip
  ```

  El contenido de ese `.zip` debe ser extraído en donde se desee (digamos que se hace en `/home/username/software`). Esta dirección debe agregarse al *path* de Python, para que el intéprete sepa donde hallar el código de este paquete. Para ello, debe ejecutarse en una terminal:

  ```
  export PYTHONPATH=$PYTHONPATH:‘‘/home/username/software/’’
  ```

  El inconveniente que tiene esto es que ese comando debe ejecutarse cada vez que se inicia una nueva sesión. Para no tener que repetirlo cada vez, puede modificarse el archivo `.bashrc` que se encuentra en el home del usuario (típicamente en `/home/username/.bashrc`). Se debe abrir ese archivo con un editor de texto cualquiera y al final agregar el comando:

  ```
  export PYTHONPATH=$PYTHONPATH:‘‘/home/username/software/’’
  ```

  Para probar si funciona, cerrar y volver a abrir la terminal y ejecutar `import ra` dentro de una sesión de Python.

Una vez instalados estos paquetes, debería ser posible ejecutar el código escrito para esta tesis. A continuación se explica cómo.

A.2. Archivos entregados y cómo ejecutar el código

Los archivos adjuntos a esta tesis se dividen en dos grandes grupos: datos y código. El primer grupo se encuentra en el directorio `data`, mientras que el segundo se encuentra en el directorio `codigo`. Ambos se descargan en un `.zip` desde la siguiente url:

`www.fing.edu.uy/~bmarenco/maestria/downloads/tesis.zip`

A.2.1. Estructura del directorio `data`

Como su nombre lo indica, el directorio `data` contiene todos los datos necesarios para ejecutar el código, además de datos de la clasificación que se proporcionan para reproducir los resultados reportados. Dentro de este directorio, se encuentran a su vez varios sub-directorios:

- `audio`: es donde se encuentran los archivos de audio utilizados para el entrenamiento y la clasificación, en formato `.wav`. Los archivos sintéticos que se usaron en el entrenamiento de las HMMs para la clasificación de patrones de repique se encuentran en `audio/train`, mientras que los archivos que se usaron para evaluar desempeño se encuentran en `audio/test`. En este último directorio hay 3 archivos:

  - `repique_jure_1`: es el archivo denominado ARS1 en esta tesis, esto es, el primer archivo de audio sintético con el que se evaluó el desempeño.
  - `repique_patrones_cambiados`: es el archivo denominado ARS2.
  - `repique_zavala_242`: es el archivo denominado ARR1, es decir, el archivo de audio real con el que se evaluó el desempeño.

  Además de los `.wav`, cada audio (tanto del directorio `train` como del `test`) tiene asociado dos archivos con extensión `.lab`. Por ejemplo, además de `repique_jure_1.wav`, existe un `repique_jure_1.lab` y un `repique_jure_1_onsets.lab`. El primero es un archivo de texto (en formato `csv`) que indica los tiempos de cada pulso del audio, necesarios para la etapa de segmentación. El segundo es una lista de tiempos con los datos de la ubicación temporal de los golpes (*onsets*) del audio correspondiente. Los archivos del directorio `train` tienen además asociado un `_groundTruth.lab` que, como su nombre lo indica, es el *ground-truth* de la clasificación (es decir, una lista de a qué clase pertenece cada compás o cada pulso).

  Los archivos utilizados para el reconocimiento de la base de piano se encuentran en `audio/train`. La nomenclatura en ese caso coincide con la de la tabla 4.6. Cada uno de estos archivos tiene tres `.csv` asociados, que indican los tiempos de cada pulso, la ubicación temporal de los golpes y el *ground-truth* de cada compás.

- `results`: en este directorio se encuentran los resultados de algunas pruebas realizadas, tanto para el piano (en `results/piano`), como para el repique (en `results/bar_clasif` para clasificación de compases y en `results/pulse_clasif` para pulsos).

  En el caso del repique, se incluye el resultado de la clasificación (esto es, la asignación a uno de los patrones rítmicos discutidos en la sección 1.2), la log-verosimilitud obtenida para todas las HMMs, y los estados ocultos recorridos, estimados con el algoritmo de Viterbi. Todos estos datos son

necesarios para usar el visualizador; en la sección A.2.2 se explica cómo generarlos (es decir, cómo realizar el proceso de entrenamiento y clasificación). Aquí se incluyen para que no sea necesario repetir el entrenamiento y la clasificación para visualizar los resultados (ya que sobre todo el entrenamiento es muy intensivo comuptacionalmente).

Para el piano, se incluyen los resultados de la clasificación para las pruebas utilizando las tomas de 4 tambores (en `results/piano/con_tomas_de_4/`) o sin utilizarlas (en `results/piano/sin_tomas_de_4/`). Además, dentro de cada uno están los resultados según las distintas frecuencias máximas de análisis (`2k/` o `4k/` para 2 kHz o 4 kHz respectivamente).

- `trainedHMMs`: como su nombre lo indica, en este directorio se encuentran las HMMs previamente entrenadas, en un archivo de extensión `.pkl`. La estructura allí es análoga a la del directorio `results`. Las HMMs se proporcionan para que no sea necesario repetir el entrenamiento, ya que ese proceso puede llevar mucho tiempo;[4] en la sección A.2.2 se explica cómo hacerlo si se desea.

- `compases` y `pulsos`: en estos directorios se encuentran las partituras de los audios ARS1, ARS2 y ARR1. Esos datos son necesarios para la visualización de resultados.

A.2.2. El directorio `codigo`: cómo ejecutar el software

En este directorio se encuentra todo el código escrito para la tesis (dentro de `codigo/python`). Esto incluye las etapas de entrenamiento y clasificación, tanto para las variantes de repique como para la base de piano, y los programas de visualización de resultados. No se pretende hacer aquí un análisis profundo de la implementación, si no que simplemente se explica qué debe ejecutarse para reproducir los resultados reportados en la tesis.

Variantes de repique

En el caso de repique, los archivos que permiten visualizar los resultados de la clasificación son `display_classification_results.py` (para la clasificación de compases) y `display_pulse_classification_results.py` (para la clasificación de pulsos) Ambos *scripts* funcionan de la misma manera: aceptan como único parámetro el nombre del archivo cuyos resultados se quieren visualizar: `repique_jure_1` para el archivo ARS1, `repique_patrones_cambiados` para el ARS2, y `repique_zavala_242` para el ARR1. Por ejemplo, si se quieren visualizar los resultados de la clasificación del pulsos para el audio ARR1, debe ejecutarse:

```
python display_pulse_classification_results.py repique_zavala_242
```

Todos los datos necesarios para visualizar la clasificación son leídos del directorio `data`.

[4] A modo de ejemplo, entrenar las HMMs para el reconocimiento de la base de piano lleva unas 5 horas en una máquina i7 con 16 Gb de RAM.

Si quiere repetirse el proceso de entrenamiento y/o clasificación para el reconocimiento de compases, los programas para hacerlo son `hmm_training.py` (para el entrenamiento) y `hmm_classification.py` (para la clasificación). Los equivalentes para pulsos son `hmm_pulse_training.py` y `hmm_pulse_classification.py`. Ninguno de estos programas acepta parámetros. Todos los parámetros de estos procesos (el tamaño de la ventana y del salto para el cálculo de los MFCCs o del *spectral flux*, la frecuencia máxima de análisis, etc.) se encuentran en el archivo `parameters.py`; debe modificarse este archivo si se desea cambiar algún parámetro.

Si quiere realizarse el proceso de entrenamiento de las HMMS, se debe ejecutar:[5]

```
python hmm_training.py.
```

Este programa reescribe en el directorio `data/trainedHMMs/` los archivos `.pkl` donde se guardan las HMMs entrenadas. Al hacer:

```
hmm_classification.py
```

se leen esos archivos y se realiza la clasificación de los audios ARS1, ARS2 y ARR1, guardando en `data/results/` los resultados.[6]

Base de piano

Para la base de piano, el visualizador de resultados se encuentra en el archivo `piano_pattern_analysis.py`. Este *script* recibe como parámetro el nombre del archivo cuya clasificación quiere visualizarse, según la nomenclatura de la tabla 4.6. Se pueden proporcionar dos parámetros adicionales:

- `-f` o `--four_drums`: la presencia de este parámetro le dice al visualizador que use las tomas con cuatro tambores; si este parámetro no aparece, las tomas de cuatro tambores no se utilizan. Esto es especialmente importante, ya que ejecutar:

  ```
  python piano_pattern_analysis.py Take_311
  ```

  causa un error, pues el archivo `Take_311` es de cuatro tambores, pero no aparece el parámetro `-f`.

- `-max_freq`: este parámetro numérico le indica al visualizador qué frecuencia máxima de análisis utilizar (en Hz). Si no aparece, se asume que la frecuencia máxima es 4 kHz.

Así, si se quieren visualizar los resultados de la clasificación para el archivo `Take_311` usando una frecuencia máxima de análisis de 2 kHz debe ejecutarse:

```
python piano_pattern_analysis.py -f -max_freq 2000 Take_311
```

[5] El ejemplo es para reconocimiento de compases; para pulsos debe ejecutarse `python hmm_pulse_training.py`.

[6] Nuevamente, el ejemplo es para reconocimiento de compases; para pulsos debe ejecutarse `python hmm_pulse_classification.py`.

El visualizador, al igual que para el caso del repique, lee los resultados de la clasificación del directorio `data`. Si se quiere volver a realizar el entrenamiento y/o la clasificación, el programa para hacerlo es `piano_recognition.py`. En este programa se puede realizar tanto el entrenamiento como la clasificación de todas las tomas de piano. Acepta varios parámetros opcionales:

- `-t` o `--do_training`: la presencia de este parámetro le indica al programa que realice el proceso de entrenamiento. Si no aparece, las cadenas entrenadas son leídas del directorio `data/trainedHMMs/piano/`.
- `-v` o `--verbose_training`: en caso de realizar el entrenamiento, que se incluya este parámetro le indica al programa que imprima datos del proceso de entrenamiento (cantidad de iteraciones de Baum-Welch, convergencia del algoritmo, etc.). Si no aparece el parámetro `-t`, `-v` no tiene efecto.
- `-f` o `--four_drums`: al igual que para el visualizador, la presencia de este parámetro le dice al programa que use las tomas con cuatro tambores.
- `-max_freq`: como en el visualizador, este parámetro numérico indica qué frecuencia máxima de análisis utilizar (en Hz). Si no aparece, se asume que la frecuencia máxima es 4 kHz.
- `-r` o `--read_scores`: indica si se deben leer de disco las log-verosimilitudes de cada compás. Estas verosimilitudes se encuentran `data/results/piano/`, en los archivos con terminación `_scores.npy`. Si no se incluye, las log-verosimilitudes se calculan con las cadenas (entrenadas en el momento o leídas de disco, según el parámetro `-t`).
- `-s` o `--save_results`: indica si se deben guardar los resultados de la clasificación, sobreescribiendo lo que hay en `data/results/piano/`.
- `-d` o `--disp_cumulative_hist`: si aparece este parámetro, se muestra el histograma acumulativo de las log-verosimilitudes del archivo de *test*, junto con el umbral que se usa para la clasificación.

Así, si se quiere realizar el proceso de entrenamiento y clasificación usando las tomas de cuatro tambores y una frecuencia máxima de 2 kHz, guardando los resultados de la clasificación para una posterior visualización, debe ejecutarse:

```
python piano_recognition.py -t -f -max_freq 2000 -s
```

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] cSounds. http://www.csounds.com/. Acceso: viernes 1 noviembre, 2019.

[2] Definición de timbre - Escuela Universitaria de Música. Disponible en: http://www.eumus.edu.uy/docentes/maggiolo/acuapu/tbr.html. Acceso: viernes 1 noviembre, 2019.

[3] LilyPond. http://lilypond.org/index.es.html. Acceso: viernes 1 noviembre, 2019.

[4] Charles Ames. The Markov process as a compositional model: a survey and tutorial. *Leonardo*, pages 175–187, 1989.

[5] Raimo Bakis. Continuous speech recognition via centisecond acoustic states. *The Journal of the Acoustical Society of America*, 59(S1):S97–S97, 1976.

[6] Leonard E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[7] Leonard E. Baum, John Alonzo Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363, 1967.

[8] Leonard E. Baum and George Sell. Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227, 1968.

[9] J Robert Beck and Stephen G Pauker. The Markov process in medical prognosis. *Medical Decision Making*, 3(4):419–458, 1983.

[10] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In Kathryn Huff and James Bergstra, editors, *Proceedings of the 14th Python in Science Conference*, pages 18 – 24, 2015.

[11] Juan José Burred and Alexander Lerch. A hierarchical approach to automatic musical genre classification. In *Proceedings of the 6th international conference on digital audio effects*, pages 8–11, 2003.

[12] Ruofeng Chen, Weibin Shen, Ajay Srinivasamurthy, and Parag Chordia. Chord recognition using duration-explicit hidden Markov models. In *ISMIR*, pages 445–450. Citeseer, 2012.

[13] Parag Chordia and Alex Rae. Tabla gyan: An artificial tabla improviser. In *Proc. of International Conference on Computational Creativity*, 2010.

[14] Parag Chordia, Avinash Sastry, and Sertan Şentürk. Predictive tabla modelling using variable-length Markov and Hidden Markov Models. *Journal of New Music Research*, 40(2):105–118, 2011.

[15] Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition*, pages 65–74. Elsevier, 1990.

[16] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[17] Arne Eigenfeldt and Philippe Pasquier. Realtime generation of harmonic progressions using controlled Markov selection. In *Proceedings of ICCC-X-Computational Creativity Conference*, 2010.

[18] Daniel P. W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab. `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`, 2005. Acceso: viernes 1 noviembre, 2019.

[19] Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[20] Mary Farbood and Bernd Schoner. Analysis and synthesis of Palestrina-style counterpoint using Markov chains. In *Proceedings of the International Computer Music Conference*, pages 471–474, 2001.

[21] Luis Ferreira. El repicado del Candombe. In *V Jornadas Argentinas de Musicología, Instituto Nacional de Musicología "Carlos Vega", Bs.As.*, 1990.

[22] Luis Ferreira. *Los Tambores del Candombe*. Colihue-Sepé Ediciones, 1997.

[23] Alexander Fischer and Volker Stahl. Database and online adaptation for improved speech recognition in car environments. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, pages 445–448. IEEE, 1999.

[24] Jonathan T Foote. Content-based retrieval of music and audio. In *Voice, Video, and Data Communications*, pages 138–147. International Society for Optics and Photonics, 1997.

[25] G. David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[26] Mark JF Gales, David Pye, and Philip C Woodland. Variance compensation within the MLLR framework for robust speech recognition and speaker adaptation. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 3, pages 1832–1835. IEEE, 1996.

[27] Olivier Gillet and Gaël Richard. Automatic labelling of tabla signals. 2003.

[28] Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.

[29] Philip N Johnson-Laird. Jazz improvisation: A theory at the computational level. *Representing musical structure, London*, pages 291–325, 1991.

[30] Biing Hwang Juang and Laurence R Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[31] Bing-Hwang Juang. Maximum-Likelihood Estimation for Mixture Multivariate Stochastic Observations of Markov Chains. *AT&T Technical Journal*, 64(6):1235–1249, 1985.

[32] Bing-Hwang Juang, Stephene Levinson, and M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Transactions on Information Theory*, 32(2):307–309, 1986.

[33] Luis Jure. ¡Perico, suba ahí! Pautación y análisis de un solo de repique de Pedro 'Perico' Gularte. In *VII Jornadas Argentinas de Musicología, Instituto Nacional de Musicología "Carlos Vega", Bs.As.*, 1992.

[34] Luis Jure. Principios generativos del toque de repique del candombe. In C. Aharonián, editor, *La música entre África y América*, pages 263–291, Montevideo, Uruguay, 2013. Centro Nacional de Documentación Musical Lauro Ayestarán.

[35] Luis Jure and Olga Picún. Los cortes de los tambores. Aspectos musicales y funcionales de las paradas en las llamadas de tambores afro montevideanos. In *VII Jornadas Argentinas de Musicología, Instituto Nacional de Musicología "Carlos Vega", Bs.As.*, 1992.

[36] Luis Jure and Martín Rocamora. Clave patterns in Uruguayan Candombe drumming. In *16th Rhythm Production and Perception Workshop (RPPW 2017), Birmingham, UK, 3 - 5 jul*, 2017.

[37] Luis Jure and Martín Rocamora. Microtiming in the rhythmic structure of candombe drumming patterns. In *Fourth International Conference on Analytical Approaches to World Music, New York, USA, 8-11 jun*, pages 1–6, 2016.

[38] David G Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[39] Anssi P Klapuri, Antti J Eronen, and Jaakko T Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):342–355, 2005.

[40] Christopher J Leggetter and Philip C Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer speech & language*, 9(2):171–185, 1995.

[41] L. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734, 1982.

[42] Beth Logan et al. Mel Frequency Cepstral Coefficients for Music Modeling. In *ISMIR*, 2000.

[43] Mıriam Luján, Carlos D Martınez, and Vicente Alabau. Evaluation of several maximum likelihood linear regression variants for language adaptation. In *Proceedings of the 6th International Language Resources and Evaluation Conference*, volume 21, 2008.

[44] L.S. Maia, P. D. de Tomaz Jr., M. Fuentes, M. Rocamora, L. W. P. Biscainho, M. V. M. da Costa, and S. Cohen. A novel database of brazilian rhythmic instruments and some experiments in computational rhythm analysis. In *2018 Audio Engineering Society Latin American Conference, 24-26 set.*, Montevideo, Uruguay, 2018.

[45] Bernardo Marenco, Magdalena Fuentes, Florencia Lanzaro, Martín Rocamora, and Alvaro Gómez. A multimodal approach for percussion music transcription from audio and video. In *Proceedings of the XX Iberoamerican Congress on Pattern Recognition (CIARP)*, 9-12 November 2015.

[46] K. D. Martin. Sound-Source Recognition: A Theory and Computational Model. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on.* PhD thesis, MIT. Cambridge, MA., 1999.

[47] Paul Masri. *Computer modelling of sound for transformation and synthesis of musical signals.* PhD thesis, University of Bristol, 1996.

[48] Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation.* Springer Science & Business Media, 2009.

[49] Leonardo Nunes, Martín Rocamora, Luis Jure, and Luiz W. P. Biscainho. Beat and downbeat tracking based on rhythmic patterns applied to the uruguayan candombe drumming. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015). Málaga, Spain*, pages 264–270, 26-30 oct 2015.

[50] Joao Lobato Oliveira, Matthew EP Davies, Fabien Gouyon, and Luís Paulo Reis. Beat tracking for multiple applications: A multi-agent system architecture with state recovery. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(10):2696–2706, 2012.

[51] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[52] François Pachet and Pierre Roy. Markov constraints: steerable generation of markov sequences. *Constraints*, 16(2):148–172, 2011.

[53] François Pachet and Pierre Roy. Imitative leadsheet generation with user constraints. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, pages 1077–1078. IOS Press, 2014.

[54] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the Web. 1999.

[55] Jouni Paulus and Anssi Klapuri. Drum sound detection in polyphonic music with hidden Markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:14, 2009.

[56] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[57] L.R. Rabiner and R.W. Schafer. Digital Processing of Speech Signals. Prentice Hall, New Jersey, 1978.

[58] Martín Rocamora. *Computational methods for percussion music analysis : The Afro-Uruguayan Candombe drumming as a case study"*. PhD thesis, Universidad de la República (Uruguay). Facultad de Ingeniería. IIE, apr 2018.

[59] Martín Rocamora and Luiz W. P. Biscainho. Modeling onset spectral features for discrimination of drum sounds. In *Proceedings of the 20th Iberoamerican Congress on Pattern Recognition (CIARP). Montevideo, Uruguay*, pages 100–107, 9-12 nov 2015.

[60] Martín Rocamora, Luis Jure, and Luiz W. P. Biscainho. Tools for detection and classification of piano drum patterns from candombe recordings. In *Proceedings of the 9th Conference on Interdisciplinary Musicology (CIM14)*, page 382–387, 4-6 December 2014.

[61] Martín Rocamora, Luis Jure, Bernardo Marenco, Magdalena Fuentes, Florencia Lanzaro, and Álvaro Gómez. An audio-visual database of Candombe performances for computational musicological studies. In *Proceedings of the II Congreso Internacional de Ciencia y Tecnología Musical*, 17-19 September 2015.

[62] X. Serra. A multicultural approach in music information research. In *International Society for Music Information Retrieval Conference (ISMIR)*, Miami, Florida, USA, 2011.

[63] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el viernes 1 noviembre, 2019.
http://iie.fing.edu.uy/