UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DETECCIÓN Y CLASIFICACIÓN DE SEÑALES SUBMARINAS CON TÉCNICAS DE *MACHINE LEARNING*

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

REINALDO CRISTIAN HOFFMANN VÁSQUEZ

PROFESORA GUÍA:
SUSANNAH BUCHAN

MIEMBROS DE LA COMISIÓN:
NÉSTOR BECERRA YOMA
RODRIGO MAHU SINCLAIR

Este trabajo ha sido financiado por Fondecyt bajo el proyecto Iniciación 11190597

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: REINALDO CRISTIAN HOFFMANN VÁSQUEZ
FECHA: 2021
PROF. GUÍA: SUSANNAH BUCHAN
PROF. CO-GUÍA: NÉSTOR BECERRA YOMA

DETECCIÓN Y CLASIFICACIÓN DE SEÑALES SUBMARINAS CON TÉCNICAS DE *MACHINE LEARNING*

En este trabajo se busca detectar y clasificar correctamente eventos presentes en grabaciones submarinas capturadas en las costas de la isla de Juan Fernández, Chile. Para la detección y clasificación se aplican técnicas de *machine learning*, en particular se utilizan modelos ocultos de Markov (HMM), redes neuronales profundas (DNN) y mezclas de gaussianas (GMM).

Todos los modelos utilizados consisten en un modelo oculto de Markov, cuyas probabilidades de observación son calculadas de dos formas distintas. Primero se prueba utilizando una mezcla de gaussianas para obtener las probabilidades y posteriormente se utiliza una red DNN en su lugar.

Se dispone de 1.381 grabaciones de aproximadamente 10 minutos, correspondiente a 236 horas totales de audio. En las grabaciones aparecen 40.857 eventos distintos de silencio. Se usan 24 clases distintas para realizar la clasificación, entre las que se tienen vocalizaciones de ballenas azules, vocalizaciones de ballenas sei, vocalizaciones de ballenas minke, sonidos provenientes de barcos, sonidos asociados a movimientos sísmicos, entre otros. La base de datos presenta un gran desbalance de clases debido a que gran parte de las grabaciones corresponde a silencio y ruido de fondo.

Para evaluar los resultados se utilizan tres métricas: el porcentaje de clasificaciones correctas total, el porcentaje de clasificaciones correctas de los eventos distintos de silencio y el porcentaje de clasificaciones correctas de los eventos sísmicos. Se calcula el *accuracy* de este evento en particular debido al interés práctico por las posibles aplicaciones de la detección y clasificación de movimientos sísmicos.

Se obtienen porcentajes de *accuracy* total de 90 % aproximadamente, tanto usando modelos HMM con redes DNN como usando modelos HMM con GMM. El porcentaje de clasificaciones correctas de los eventos sísmicos usando redes DNN es muy alto, superando 97 % en varios experimentos. Por otra parte, el porcentaje de clasificaciones correctas de todas las etiquetas distintas de silencio son menores, fluctuando entre 50 % y 30 % en la mayoría de los experimentos.

Se concluye que el gran desbalance de clases, con la mayoría de los $frames$ correspondientes a silencio, afecta la clasificación de los demás eventos. Los modelos con mezcla de gaussianas entregan un mayor porcentaje de clasificaciones correctas de las etiquetas distintas de silencio y modelos con DNN clasifican de mejor manera los eventos sísmicos, por lo que se concluye que el mejor modelo depende de cuál de estos factores resulta más importante para la aplicación en particular.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introduccion

1.1. Antecedentes Generales

Los avances tecnológicos realizados en las últimas décadas han resultado en la generación de grandes volúmenes de datos en diversas áreas e industrias. Los desarrollos en sensores, por ejemplo, han bajado dramáticamente los precios de estos dispositivos, lo que ha masificado su uso, resultando en gran cantidad de datos captados. Así mismo, avances en almacenamiento de datos han permitido guardar estos datos generados a precios relativamente bajos, de manera que estos datos pueden utilizados posteriormente. Paralelamente, se han hecho grandes progresos en el desarrollo de algoritmos eficientes y de procesadores capaces de ejecutarlos, por lo que la capacidad de procesar y utilizar datos ha aumentado a un ritmo acelerado.

Así, en las últimas décadas la cantidad de datos generados y las capacidades para utilizarlos han llegado a niveles inesperados, dando lugar a nuevas profesiones enfocadas particularmente en explotar esta nueva posibilidad. Las empresas y gobiernos no tardaron demasiado en notar las grandes ventajas y aplicaciones que se pueden obtener a partir de la correcta utilización de estos grandes volúmenes de datos y así profesiones como Data Scientist, especialistas en Big Data y especialistas en Machine Learning han ganado gran importancia, siendo parte de los roles con mejores perspectivas apoyados por la revolución informática.

Para aprovechar de la mejor manera posible los datos disponibles, se han ido perfeccionando continuamente los algoritmos y modelos computacionales implementados. Se han desarrollado diversos modelos, como Support Vector Machines, Random Forest, Redes Neuronales Convolucionales, LSTM, entre otros. Debido a que actualmente la cantidad de datos es muy grande, se tiene mucha información para entrenar los modelos computaciones, por lo que ha aumentado el interés por algoritmos no supervisados y algoritmos que mejoren su rendimiento continuamente al aumentar la cantidad de datos para entrenar.

El interés en algoritmos no supervisados surge debido a que estos no requieren de datos etiquetados para el entrenamiento, lo que es de vital importancia cuando se tienen grandes volúmenes de datos y no se pueden etiquetar para entrenar por razones prácticas. Por otra parte, no todos los modelos mejoran continuamente su rendimiento al aumentar la cantidad de datos de entrenamiento. La gran mayoría de los modelos están limitados en la complejidad de las relaciones que pueden modelar y el aumento de datos es insuficiente para superar esa limitación, por lo que dejan de mejorar su $performance$ al aumentar el número de datos

disponibles. Así, estos algoritmos no sacan el máximo provecho de la gran cantidad de datos disponibles y esto ha llevado el interés a modelos como las Redes Neuronales Profundas y el Deep Learning, que modelan relaciones más complejas al aumentar la cantidad de datos.

1.2. Identificación y Formulación del Problema

En la actualidad se tienen numerosos datos en forma de grabaciones acústicas obtenidos usando hidrófonos ubicados en el fondo del mar. En particular, una de las técnicas utilizadas es el monitoreo acústico pasivo PAM (*Passive Acoustic Monitoring*) con el cual se da seguimiento a la presencia temporal y espacial de vocalizaciones de mamíferos acuáticos. Esta técnica permite detectar por medio de hidrófonos vocalizaciones de baja frecuencia de ballenas y otros sonidos a decenas de kilómetros de distancia de la fuente.

Para obtener información relevante de estas grabaciones, tradicionalmente se obtienen espectrogramas que luego son analizados por expertos. Esta metodología requiere que los expertos recorran cada grabación y etiqueten manualmente los objetos detectados. Este sistema resulta muy costoso en términos de tiempo y esfuerzo por parte de los analistas. Por esta razón, se ha explorado la aplicación de técnicas computacionales para realizar la detección y clasificación de objetos presentes en las grabaciones. Así, se comenzó a utilizar técnicas de correlación cruzada para la detección de eventos, midiendo la similitud de las señales con templados estándar de los eventos a detectar y otras técnicas similares. Las técnicas computacionales disponibles han avanzado significativamente y en la actualidad se disponen de variados modelos para problemas de detección y clasificación en general. En este trabajo se busca aplicar técnicas actuales de *machine learning* para obtener modelos capaces de detectar y clasificar los eventos presentes en las grabaciones. En particular, se utilizan modelos ocultos de Markov, con dos formas distintas de calcular las probabilidades de observación de los modelos: usando un modelo de mezcla de gaussiana o usando una red neuronal profunda.

En las grabaciones se dispone de múltiples eventos distintos. Se tienen cantos y vocalizaciones de varias especies de ballenas (ballenas azules, ballenas sei, ballenas fin y ballenas minke), se tienen sonidos provenientes de barcos, sonidos resultantes de los movimientos de las placas tectónicas, entre otros. También aparece ruido de fondo y ruido proveniente de la plataforma de detección. Dada la gran variedad de eventos, se requiere entrenar modelos capaces de clasificar un gran número de clases distintas.

En resumen, con este trabajo se busca realizar la detección y clasificación de objetos presentes en las grabaciones acústicas usando modelos ocultos de Markov, junto con modelos de mezclas de gaussianas o con redes DNN. Con esto se busca eliminar la necesidad de que expertos etiqueten manualmente los objetos en las grabaciones, aun cuando se tiene grabaciones con un gran número de eventos distintos.

1.3. Objetivos del Trabajo de Título

El objetivo de este trabajo es obtener porcentajes de clasificaciones correctas (*accuracy*) similares o superiores a los obtenidos usando métodos tradicionales, sin aumentar el porcen-

taje de falsos negativos en la clasificación, de manera que su ventaja en tiempo y esfuerzo lo haga una opción atractiva frente a estos métodos.

Puesto que ya existen trabajos en la literatura que trabajan en problemas similares usando técnicas de *machine learning*, resulta necesario obtener resultados de clasificaciones correctas comparables con los obtenidos en estos trabajos. Se busca determinar la utilidad de las técnicas usadas en el trabajo para abordar este tipo de problema. Los resultados obtenidos permiten la comparación con otras técnicas de *machine learning* para la detección y clasificación de objetos en grabaciones submarinas.

Otro objetivo del trabajo es ofrecer un modelo capaz de detectar y clasificar un gran numero de distintos eventos. La idea es tener un sistema que sea capaz de clasificar no sólo vocalizaciones de ballenas, sino también ruidos de barcos, movimientos de placas tectónicas, etc. Así, este trabajo muestra la viabilidad de entrenar modelos para clasificar un gran numero de clases distintas.

Se explorará la posibilidad de extraer información relevante asociada a movimientos sísmicos a partir de las grabaciones, con la intención de detectar movimientos telúricos antes de que lleguen al continente. Las detecciones podrían comunicarse directamente a centros sismológicos, lo que podría ayudar a alertar la presencia de sismos con mayor rapidez.

1.4. Alcances del Proyecto

Como se mencionó anteriormente, el trabajo realizado está enfocado en obtener buenos resultados de clasificación utilizando un número alto de clases distintas. Se busca aportar en el avance de las técnicas usadas para la clasificación de sonidos submarinos, por lo que el trabajo se limita a mostrar el porcentaje de clasificaciones correctas que es posible obtener con los modelos propuestos. Así, el trabajo no se extiende a implementar los modelos creados en terreno para realizar clasificaciones y ser usado en la práctica.

Este es un paso que se puede realizar pero que requiere de trabajo adicional como, por ejemplo, determinar cómo ingresar nuevos datos de manera automatizada al modelo. Considerando esto, los modelos implementados no están diseñados pensando en una aplicación en tiempo real de clasificación, por lo que el tiempo de ejecución de los algoritmos no es una preocupación al momento de obtener resultados. En caso de querer extender los modelos creados para aplicaciones prácticas, sin duda estos factores tendrían que ser considerados para la selección del modelo óptimo para la clasificación detección de objetos submarinos.

1.5. Justificación del Proyecto

1.5.1. Relevancia del Problema

Múltiples arreglos de hidrófonos fueron instalados en el fondo del mar por Organización del Tratado de Prohibición Completa de los Ensayos Nucleares (CTBTO) con la intención de

monitorear los mares y garantizar el tratado de prohibición de ensayos nucleares se cumpla. Hoy en día, los datos obtenidos con estos hidrófonos pueden ser utilizados para aplicaciones adicionales, una de las más interesantes es el monitoreo de la población de animales marinos por medio de las vocalizaciones captadas por los arreglos.

Con la correcta utilización de las grabaciones, es posible estimar el tamaño de las poblaciones de distintos animales marinos y estudiar la distribución temporal y espacial de estos. Esta información es de vital importancia para tener constancia de los efectos de los distintos cambios en los ambientes naturales de estos seres vivos. Así, estos datos son una herramienta fundamental para conocer la situación actual de las distintas especies en esfuerzos para proteger y conservar la biodiversidad de los mares y océanos de nuestro planeta.

Más aún, entre los sonidos captados por los hidrófonos se encuentran sonidos generados por movimientos telúricos, generados por el movimiento de las placas tectónicas. Una posibilidad es captar estos sonidos por medio de los arreglos de hidrófonos y alertar prematuramente a las autoridades de costas cercanas de posibles sismos. Esta herramienta requiere ser capaces de desarrollar un sistema en tiempo real, lo que sale del alcance de este proyecto, pero los resultados obtenidos con el trabajo pueden ofrecer motivación y una base para trabajos posteriores que si funcionen en tiempo real.

1.5.2. Laboratorio de Procesamiento y Transmisión de Voz

El Laboratorio de Procesamiento y Transmisión de Voz de la Universidad de Chile (LPTV), fue creado el año 2000 por el Profesor Néstor Becerra Yoma, Ph.D., como parte del proyecto "Procesamiento Robusto de Patrones Acústicos para Aplicaciones en telefonía e Internet", financiado por Conicyt/Foncecyt, Chile. Hoy en día el laboratorio se enfoca en proyectos relacionados con tecnologías de voz, con *Quality of Service* (*QoS*) en Internet y la aplicación de estas tecnologías en ingeniería. En el laboratorio también se han desarrollado trabajos de procesamiento de señales en diversas áreas, como en detección de ballenas, en sismología y en astronomía.

El laboratorio ha mantenido colaboración con importantes universidades y centros de investigación como University of Southern California, University of Colorado, University of Edinburgh, entre otros. Muchos de los resultados de investigación obtenidos en el laboratorio han sido presentados en revistas y conferencias internacionales prestigiosas en el campo de tecnologías de voz, con múltiples *papers* publicados en el área.

En el LPTV se tiene amplia experiencia en problemas de clasificación y detección como el que se aborda en este trabajo, por lo que resulta un ambiente ideal para desarrollar este proyecto. Ya se han realizado trabajos similares en el pasado y la mayoría de los integrantes del laboratorio tienen experiencia en este tipo de problemas.

El trabajo realizado se lleva a cabo dentro del marco del proyecto Fondecyt Iniciación 11190597 dirigido por la Dra. Susannah Buchan y fue desarrollado dentro del Laboratorio de Procesamiento y Transmisión de Voz de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. Para realizar el trabajo se hace uso de CPU con procesador modelo Intel Core i7-4790 de 3.6 GHz, con tarjeta de vídeo GeForce GTX 980, disponible en

el laboratorio.

1.5.3. Agradecimientos

Se agradece a la Organización del Tratado de Prohibición Completa de los Ensayos Nucleares (CTBTO) por suministrar las grabaciones utilizadas en este trabajo. Los datos fueron obtenidos por medio de un acuerdo entre la Universidad de Concepción, la Universidad de Washington y la CTBTO.

Se agradece al Fondo Nacional de Desarrollo Científico y Tecnológico (Fondecyt) por el financiamiento de la investigación realizada en este proyecto. Este trabajo es parte del proyecto Fondecyt Iniciación 11190597 dirigido por la Dra. Susannah Buchan, Universidad de Chile.

1.6. Estado del Arte

El procedimiento tradicional de detección y clasificación de objetos en grabaciones submarinas consiste en obtener espectrogramas a partir de las grabaciones y luego analizar cada grabación, etiquetando manualmente los objetos detectados. Este proceso debe ser realizado por expertos y requiere de muchas horas de trabajo.

Como solución a este problema, en la literatura se ha intentado usar correlación cruzada de espectrogramas (*Spectrogram Cross-Correlation*), que es una técnica que mide la similitud de una señal de entrada con templados de distintos tipos de cantos. Esta técnica permite detectar cantos estereotípicos de ballenas, cuando estos se asemejan a algún templado (Stafford et al. 1999[1]; Mellinger and Clark 2000[2]; Samaran et al. 2013[3]; Buchan et al. 2015[4]). En la práctica este método sigue requiriendo tiempo de análisis para detectar falsos positivos y falsos negativos.

Un método empleado en la actualidad que hace uso de herramientas computacionales es el *Pitch Tracking*. Con esta técnica se caracteriza la variación temporal de la frecuencia dominante de la vocalización y se realiza la clasificación basado en las características de este seguimiento de la frecuencia dominante. Este método fue introducido en el año 2011 por Baumgartner y Mussoline (Baumgartner et Mussoline 2011[5]), donde se utiliza *QDFE* (*Quadratic Discriminant Function Analysis*) para realizar la clasificación.

Otra solución propuesta en la literatura es el uso de modelos ocultos de Markov (HMM) para modelar los cantos de ballenas presentes en grabaciones acústicas y detectar la presencia de vocalizaciones de ballenas azules en estas. El sistema propuesto produce un porcentaje de clasificaciones correctas (*accuracy*) de 85.3 % (Buchan et al. 2019[6]). Este trabajo se limita solamente a la detección y clasificación de cantos de ballenas azules.

En Garcia et al. 2020[7] se utilizan cinco métodos de *machine learning* distintos para la clasificación de vocalizaciones de ballenas Fin. Los métodos implementados son *Support Vector Machines* (SVM), Redes Neuronales Convolucionales (CNN), *Long Short-Term Memory* (LSTM), arboles de decisión y regresión logística. La clasificación se realiza para 6 clases

distintas y se obtienen valores de *accuracy* altos, con porcentajes mayores a 95 %.

En la literatura se ha intentado realizar clasificaciones binarias usando vocalizaciones de ballenas Jorobadas. En Kalkhoran et al. 2019[8] se utilizan grabaciones obtenidas con hidrófonos ubicados en el Golfo de Maine, clasificando las vocalizaciones en dos clases: cantos y no cantos. En el trabajo se prueban tres métodos distintos para la clasificación: *Support Vector Machines*, Redes Neuronales y *Gaussian Naive Bayes*. Como vector de características para representar las grabaciones se prueba con *Mel-frecuency cepstrum* (MFCC) y/o *Power Stectrogram Density* (PSD). El mejor resultado de clasificaciones correctas obtenido es 94 %, usando SVM como modelo y MFCC como vectores de $features$.

En Pace et al. 2010[9] se propone una nueva forma de modelar las vocalizaciones de ballenas, en este caso de ballenas jorobadas. Las vocalizaciones de ballenas están formadas por unidades y generalmente se generan modelos computacionales en base a estas unidades. En este trabajo se divide las unidades en subunidades cunado existen cambios en la frecuencia fundamental dentro de una unidad y se modela en base a estas subunidades. La hipótesis es que estas subunidades son menos variables que las unidades, por lo que es más fácil para el modelo captarlas. Como datos utilizan grabaciones acústicas de la costa de Madagascar y a partir de estas extraen vectores de características usando *Linear Prediction filter Coefficients*, MFCC y *Ceptral Coefficients*. El modelo usado para la clasificación es *K-Means Clustering* con 18 clases distintas (18 subunidades) y obtienen aproximadamente 70 % de *accuracy*. Los resultados obtenidos modelando con subunidades son mejores a los obtenidos modelando con unidades en este trabajo.

Se ha intentado también desarrollar métodos de clasificación genéricos para sonidos provenientes de animales marinos. En Malfante et al. 2018[10] se pone énfasis en la creación de una representación robusta de las grabaciones, proponiendo vectores de características de 84 componentes, extraídos del dominio del tiempo, del dominio de la frecuencia (MFCC) y del dominio ceptral (capta las propiedades periódicas de la señal). La idea es que este amplio vector de $features$ puede ser utilizado de forma general para *datasets* de distinta naturaleza. En el trabajo se utilizan datos captados por *Passive Acustic Monitoring* en las costas de Francia en el Mar Mediterráneo en 2014. Como modelos computaciones utilizan *Support Vector Machines* y *Random Forest* para clasificar los sonidos en seis clases distintas. Se obtienen altos porcentajes de clasificaciones correctas usando estos vectores de $features$: 95.3 % usando *Random Forest* y 95 % usando *Support Vector Machines*.

El uso de técnicas de *machine learning* no se ha limitado a realizar clasificaciones de eventos y vocalizaciones. Por ejemplo, en Shabangu et al. 2020[11] se utiliza un modelo *Random Forest* (RF) para determinar la influencia de distintas variables en la ocurrencia de vocalizaciones de ballena. Los datos usados corresponden a grabaciones provenientes del mar de Weddell en el océano Antártico, en las que aparecen llamadas tipo Z y tipo D de ballenas azules antárticas y pulsos de $20Hz$ de ballenas fin. Una vez detectadas las vocalizaciones usando correlación cruzada con templados, se utiliza RF para investigar la influencia de tres variables (hora del día, distancia al casquete polar y mes del año) en la ocurrencia de vocalizaciones de ballena azul antártica y ballena fin. Usando coeficientes de Gini para determinar la importancia relativa de cada variable en el modelo RF, llegaron a la conclusión de que el mes del año es el predictor más importante, seguido por la distancia al casquete

polar. La hora del día no resulta ser un predictor relevante de la ocurrencia de vocalizaciones.

Por otra parte, el uso de modelos ocultos de Markov (HMM) en aplicaciones con señales acústicas ha sido ampliamente explorado debido a los buenos resultados que entrega. En Moreira et al. 2020[12] se utilizan señales acústicas para determinar el gesto realizado por el usuario, clasificándolas usando HMM. Se captura el sonido producido por el dedo del usuario al deslizarse por una superficie (de manera similar a una pantalla *touch*, pero usando sonido) utilizando un estetoscopio y un micrófono. Se clasifican las señales generadas en tres gestos distintos: circular, triangular y cuadrado. La aplicación está diseñada para ser implementada en la práctica y se compara los resultados de clasificación obtenidos usando un modelo HMM con los resultados usando una red neuronal. Los mejores resultados se obtienen usando HMM, con un porcentaje de *accuracy* de 90 % y tiempos de ejecución menores.

Capítulo 2

Marco Teórico

2.1. Hidden Markov Model

Un modelo de Markov es un proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. Definiendo la secuencia de variables aleatorias como $X = \{X_1, \ldots, X_T\}$ y definiendo el espacio de estados como $S = \{S_1, \ldots, S_N\}$ (posibles valores que las variables aleatorias pueden tomar), para que un proceso sea considerado como de Markov debe entonces cumplir con las siguientes condiciones o propiedades:

Horizonte limitado:

$$P(X_{t+1} = S_k | X_1, ..., X_t) = P(X_{t+1} = S_k | X_t)$$

Esta propiedad significa que la probabilidad de que en el tiempo $t + 1$ se tenga el estado S_k depende sólo del valor que tomó en el tiempo anterior t y no depende directamente de tiempos anteriores a t.

Invariante en el tiempo:

$$P(X_{t+1} = S_k | X_1, ..., X_t) = P(X_2 = S_k | X_1)$$

Esta propiedad quiere decir que la dependencia entre el valor en el tiempo $t + 1$ y t (para cualquier valor de t) es la misma que entre el valor en el tiempo $t = 2$ y $t = 1$, o sea, que la función de probabilidad no cambia en el tiempo.

Un modelo oculto de Markov es un modelo donde no se puede observar directamente los estados del sistema (*hidden states*), pero se asume que actúan de acuerdo a las propiedades de Markov. Aquí, se asume que existen otras variables que dependen de los estados ocultos y que si son observables. La idea es inferir los estados ocultos a partir de la observación de estos eventos dependientes que si se pueden medir [13]. La definición formal del HMM es la siguiente:

Un modelo oculto de Markov queda definido por la tupla $M = \{S, \sum, A, B, \Pi\}$ donde:

- S es un conjunto finito de estados, conocido como espacio de estados.

- $\sum$ es un conjunto finito de símbolos, que contiene los posibles símbolos a observar.
- A es la matriz de probabilidades de transición entre estados de S. El elemento $A[i, j]$ queda definido como $\mathrm{P}(X_{t+1} = S_j | X_t = S_i)$. En palabras, $A[i, j]$ corresponde a la probabilidad de pasar al estado S_j estando en el estado S_i.
- B es la matriz de probabilidad de emisión de símbolos de $\sum$. El elemento $B[j, k]$ queda definido como $\mathrm{P}(O_t = \mathrm{k} | X_t = S_j)$. O sea, la probabilidad de observar el símbolo k estando en el estado S_j. Las probabilidades de emisión de símbolos se conocen también como probabilidades de observación.
- Π es el vector de probabilidades iniciales. $\Pi[i]$ corresponde a la expresión $\mathrm{P}(X_1 = S_i)$. Contiene las probabilidades de comenzar la secuencia en cada posible estado.

En las definiciones dadas $X = \{X_1, ..., X_{T+1}\}$ corresponde a una secuencia de estados donde $X_t : S \rightarrow \{1, ..., N\}$. Por su parte, $O = \{O_1, ..., O_T\}$ denota una secuencia de observaciones, con $O_t \in \sum$.

Para explicar claramente la definición teórica anterior, consideremos el siguiente ejemplo clásico. Supongamos que una persona no sabe que estación del año es (invierno o verano para simplificar) y lo único que puede observar es el tiempo diario (soleado o nublado). En base a las observaciones del tiempo, quiere determinar que estación del año es y para esto construye un modelo oculto de Markov como el presentado en la figura 2.1. En este ejemplo, los estados del modelo son $S = \{S_1, S_2\} = \{verano, invierno\}$ y los símbolos observables son $\Sigma = \{soleado, nublado\}$. Las probabilidades iniciales corresponden a $\Pi = [0.6, 0.4]$, o sea, hay un 60 % de probabilidad de que se parta en el estado *verano* y 40 % de que se parta en *invierno*. Las probabilidades de transición están dadas por la matriz:

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.8 & 0.2 \end{bmatrix}$$

Por lo que, por ejemplo, la probabilidad de pasar al estado *invierno* estando actualmente en el estado *verano* es de 10 %. Finalmente, las probabilidades de emisión de símbolos (también denominadas probabilidades de observación) están denotadas con líneas entrecortadas en la figura 2.1. Considerando estos valores, la matriz B queda definida como:

$$B = \begin{bmatrix} 0.95 & 0.05 \\ 0.15 & 0.85 \end{bmatrix}$$

En palabras, la probabilidad de observar un día *soleado* estando en *verano* es 95 % y la probabilidad de observar un día *soleado* estando en *invierno* es 15 %. Por otra parte, la probabilidad de observar un día *nublado* estando en *verano* es 5 % y la probabilidad de observar un día *nublado* estando en *invierno* es 85 %.

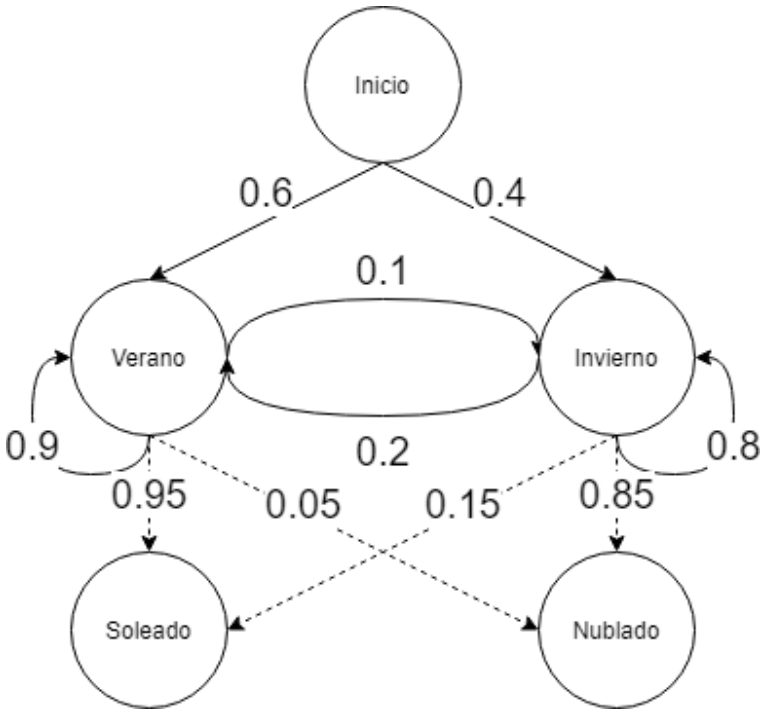

Figura 2.1: Ejemplo de modelo oculto de Markov

Existen tres problemas fundamentales relacionados con los modelos ocultos de Markov: el problema de evaluación, el problema de decodificación y el problema de aprendizaje.

El problema de evaluación corresponde a determinar que HMM es más probable que haya generado una cierta secuencia de observaciones, cuando se tienen varios modelos HMM. Aquí se busca el modelo HMM M que maximiza la probabilidad P($O|M$), con la secuencia de observaciones $O = \{O_1, ..., O_T\}$. Para calcular esta probabilidad de manera eficiente generalmente se utiliza el algoritmo $forward$.

El problema de la decodificación corresponde a determinar la secuencia de estados S más probables, dado un modelo HMM M y una secuencia de observaciones O. El criterio más utilizado consiste en elegir la secuencia de estados que maximiza la probabilidad P($S|O$, M). Para resolver eficientemente este problema, generalmente se aplica el Algoritmo de Viterbi.

El problema del aprendizaje corresponde a ajustar los parámetros del modelo HMM a entrenar de forma que se maximice la probabilidad P($O|M$), dada la secuencia de observaciones $O = \{O_1, ..., O_T\}$. Existen varios algoritmos conocidos para resolver este problema, uno de los más utilizados es el algoritmo Baum-Welch.

2.1.1. Algoritmo de Viterbi

Supongamos que se tiene un modelo HMM ya definido M y se observa la secuencia $O = \{O_1, ..., O_T\}$. Para hacer uso del modelo resulta necesario saber cuál es la secuencia de estados más probable dada esta secuencia de observaciones y el modelo HMM, ya que esta secuencia corresponde a la mejor predicción en base al modelo disponible.

Aquí, el criterio consiste en elegir la secuencia de estados $S = \{S_1, ..., S_T\}$ que maximiza la probabilidad P($S \mid O, M$). Para resolver eficientemente este problema, se aplica generalmente el algoritmo de Viterbi[13] que basa su cálculo en la variable $\delta_t(i)$ definida como:

$$\delta_t(i) = \max_{X_1,..X_{t-1}} P(X_1, ..., X_{t-1}, X_t = i, O_1, ..., O_t|M)$$

En palabras, $\delta_t(i)$ corresponde a la probabilidad del camino más probable desde tiempo 1 hasta el tiempo t, considerando sólo caminos que concuerdan con las primeras t observaciones y llegan finalmente al estado S_i.

La ventaja de definir $\delta_t(i)$ es que esta variable se puede calcular iterativamente como:

$$\delta_t(i) = B[i, O_t] \max_{1 \leq j \leq N}(\delta_{t-1}(j)A[j, i])$$

Donde A es la matriz de probabilidades de transición del modelo M.

Para determinar el camino de estados más probable, primero se calcula iterativamente $\delta_T(i)$ para todos los posibles estados finales ($\forall$i con $1 \leq$ i $\leq N$, donde N es el número de posibles estados), iterando desde $\delta_1(j) = B[j, O_1] \cdot \Pi[j]$. La matriz B corresponde a la matriz de probabilidad de emisión de símbolos y el vector Π corresponde a las probabilidades iniciales del modelo M.

Una vez se conocen los $\delta_T(i)$ $\forall i$, se extrae el último estado de la secuencia más probable como:

$$X_T = S_{i^*} = \arg \max_{1 \leq i \leq N} \delta_T(i)$$

O sea, el estado final más probable, al comparar el $\delta_T(i)$ de cada estado S_i. Una vez obtenido el último estado más probable, se determina el resto de la cadena recuperando la secuencia de estados usada para calcular $\delta_T(i^*)$, obteniendo con esto la secuencia de estados $X = \{X_1, ..., X_T\}$ final correspondiente a la mejor predicción usando este modelo.

Para aclarar la implementación de este importante algoritmo tomemos como ejemplo el modelo HMM de la figura 2.1. Consideremos que la persona toma medición solo dos días seguidos, obteniendo la secuencia de observaciones $O = \{soleado, nublado\}$ y decide utilizar el algoritmo de Viterbi para determinar la estación del año en que se encuentra. Recordemos que los estados son $S_1 = verano$ y $S_2 = invierno$ y las observaciones posibles son $O_1 = soleado$ y $O_2 = nublado$. Entonces el algoritmo se aplica como sigue:

$$\delta_1(1) = B[1, 1] \cdot \Pi[1] = 0.95 \cdot 0.6 = 0.39$$

$$\delta_1(2) = B[2, 1] \cdot \Pi[2] = 0.15 \cdot 0.4 = 0.06$$

En base a $\delta_{t=1}(1)$ y $\delta_{t=1}(2)$ se puede calcular $\delta_{t=2}(1)$ y $\delta_{t=2}(2)$ como sigue:

$$\delta_2(1) = B[1, 2] \cdot \max(\delta_1(1) \cdot A[1, 1], \delta_1(2) \cdot A[2, 1])$$

$$\delta_2(1) = 0.05 \cdot \max(0.39 \cdot 0.9, 0.06 \cdot 0.2) = 0.05 \cdot \max(0.351, 0.012) = 0.01755$$

$$\delta_2(2) = B[2,2] \cdot \max(\delta_1(1) \cdot A[2,1], \delta_1(2) \cdot A[2,2])$$

$$\delta_2(2) = 0.85 \cdot \max(0.39 \cdot 0.1, 0.06 \cdot 0.8) = 0.85 \cdot \max(0.039, 0.048) = 0.0408$$

Finalmente se aplica el criterio:

$$X_T = S_{i^*} = \arg \max_{1 \leq i \leq N} \delta_T(i)$$

En este caso, se tiene $\delta_2(1) = 0.01755$ y $\delta_2(2) = 0.0408$, por lo que $X_{t=2} = S_2 = invierno$ y reconstruyendo la secuencia máxima se tiene $X_{t=1} = S_2 = invierno$, por lo que la secuencia de estaciones más probable considerando la secuencia de observaciones $\{soleado, nublado\}$ es $\{invierno, invierno\}$. Así, en base al modelo HMM de la figura 2.1 la persona sabe que la mejor predicción de en qué estación se encuentra es invierno.

2.2. Filtro Triangular de Frecuencias

Generalmente al trabajar con señales usando redes neuronales u otros métodos, la señal no se ingresa directamente al modelo y se requiere extraer un vector de características ($features$) a partir de esta. Existen numerosos métodos para obtener este vector de $features$ a partir de una señal y uno de ellos es utilizar un banco de filtros. Aquí, el primer paso es traspasar la señal del domino temporal al dominio de la frecuencia, puesto que usualmente en este dominio se encuentra la información más relevante.

Una vez la señal está en el dominio de la frecuencia, se toman varios filtros centrados en distintas frecuencias y se arma un vector de características a partir de la componente de la señal en cada filtro, por lo que si se tienen N filtros, se creará un vector de N características. Cada filtro individual toma en consideración solo una parte del rango de las frecuencias y al aplicarse sobre una señal entrega cuanta energía de la señal está en ese rango.

La forma de los filtros puede variar, en la figura 2.2 se presenta un banco de filtros de forma triangular, con distintos anchos de banda.

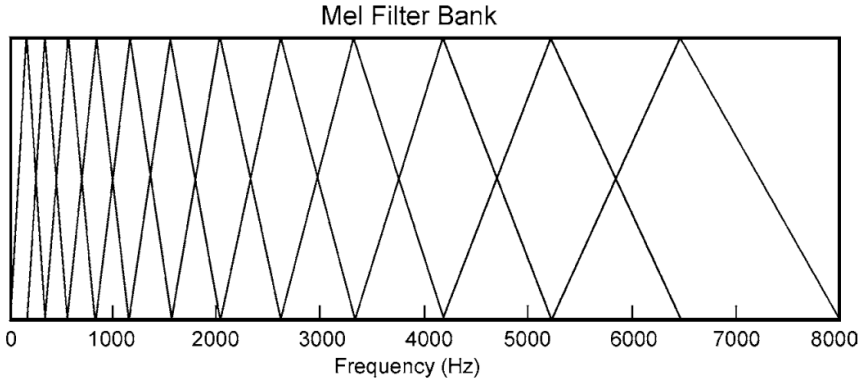

Figura 2.2: Banco de filtros en la escala de Mel

El filtro de la figura 2.2 corresponde a un banco de filtros en la escala de Mel[14]. Como se puede ver, se tiene una mayor densidad de filtros en el rango bajo de frecuencias y a

medida que se avanza en el espectro se tiene menor cantidad de filtros. Esta distribución se implementa generalmente en proyectos que trabajan con grabaciones humanas, buscando replicar la percepción humana del sonido, que es no lineal. Nuestros oídos discriminan de mejor manera en el rango de frecuencias bajas en comparación a rango de frecuencias altas y esto se puede imitar con un banco de filtros como el de la figura 2.2, que entrega una mejor resolución en el rango bajo de frecuencias.

Resulta importante mencionar que las características del banco de filtro a utilizar dependen del problema que se está abordando. Es posible acotar o extender el rango total de frecuencias, implementar mayor densidad de filtros en la región superior del espectro si se requiere mayor definición en ese rango, etc.

2.3. Gaussian Mixture Model

Un modelo de mezcla de gaussianas es un modelo probabilístico que asume que los datos son generados a partir de una mezcla de un número finito de distribuciones gaussianas, donde los parámetros de las gaussianas son desconocidos y se deben ajustar usando los datos conocidos. En lenguaje matemático un GMM es definido como:

$$P(X|\lambda) = \sum_{i=1}^{M} w_i g(X, \mu_i, \Sigma_i)$$

Donde X es un vector de datos o de características de D dimensiones, M es el número de gaussianas consideradas, w_i son los pesos de cada gaussiana (se cumple que $\Sigma_{i=1}^{M} w_i = 1$). $g(X, \mu_i, \Sigma_i)$ corresponde a la función de probabilidad de la gaussiana $i-ésima$, cada una de la forma:

$$g(X, \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(X-\mu_i)'|\Sigma|^{-1}(X-\mu_i)\right\}$$

Donde μ_i es el vector de medias de la gaussiana $i-ésima$ (de dimensión D al igual que los vectores de datos) y Σ_i es la matriz de covarianza de la gaussiana $i-ésima$. Así, el modelo GMM completo queda parametrizado por los vectores de medias, las matrices de covarianza y los pesos de cada una de las M gaussianas.

Estos modelos no requieren saber a qué gaussianas cada vector de características pertenece. En el entrenamiento el modelo va aprendiendo a diferenciar cada gaussiana y va aprendiendo la pertenencia de los datos a cada una de estas, por lo que se considera una forma de aprendizaje no supervisado[15].

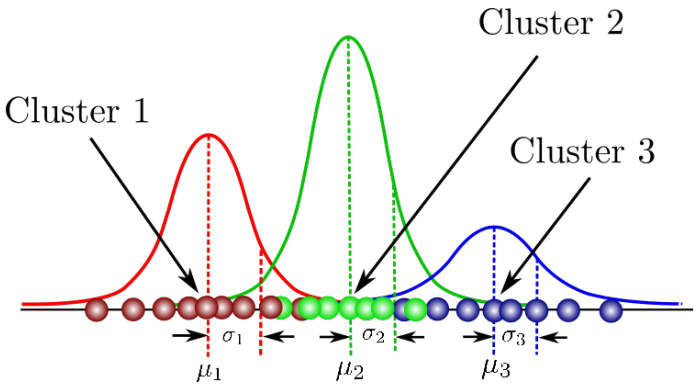

Figura 2.3: Ejemplo de modelo GMM unidimensional

En la figura 2.3 se muestra un modelo GMM de ejemplo, en este caso se tienen datos unidimensionales, o sea, $D = 1$ y se consideran 3 gaussianas, por lo que $M = 3$. Cada gaussiana está definida por su media μ_i y su varianza Σ_i. Se puede ver en la figura como estas tres gaussianas definen *clusters* entre los datos de ejemplo.

Cada gaussiana puede ser considerada como un subconjunto de los datos con cierta semejanza entre si, por lo que este método se puede utilizar para realizar *clustering*, siendo similar a *K-Means clustering* pero más robusto, ya que permite definir la pertenencia de cada dato a un *cluster*/gaussiana como una probabilidad, capturando incertidumbre en la pertenencia. Otorga una asignación más suave en comparación a la asignación binaria de *K-Means clustering*, lo que permite un mejor modelamiento de distintos tipos de datos.

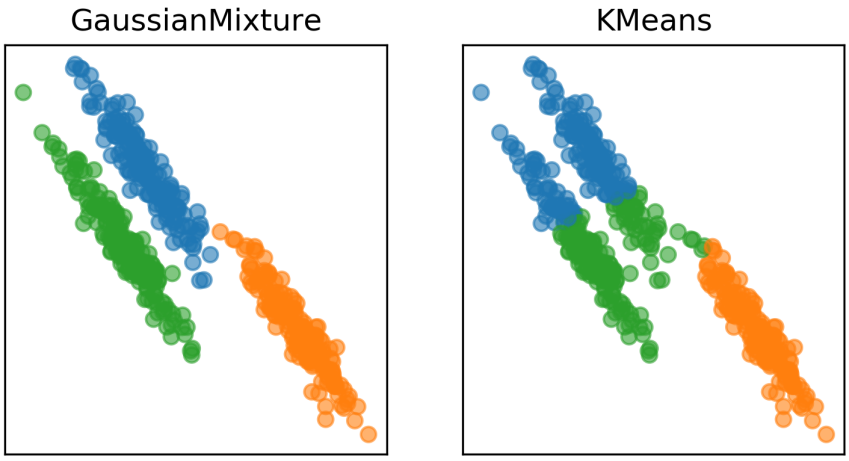

Figura 2.4: Comparación *clustering* usando GMM y usando $k - means$

En la figura 2.4 se muestra un problema de *clustering* de ejemplo que ilustra las ventajas de usar GMM sobre *K-Means*. Se puede ver como en un problema con *clusters* de forma elíptica GMM logra identificar correctamente los subconjuntos y *K-Means* no logra diferenciarlos completamente.

En el proceso de entrenamiento de un GMM se desea encontrar los parámetros que calzan mejor con la distribución de los vectores de características de los datos para entrenar.

Una forma ampliamente utilizada es el método de máxima verosimilitud (*Maximum Likelihood*). Con ML se busca maximizar la probabilidad del modelo GMM dados los datos de entrenamiento $X = \{X_1, ..., X_T\}$. Asumiendo independencia entre los vectores de datos, la probabilidad se puede escribir como:

$$P(X|\lambda) = \prod_{t=1}^{T} P(X_t|\lambda)$$

.

Puesto que la expresión es una función no linear de los parámetros λ, no es posible maximizar directamente la función para obtener los parámetros λ óptimos. Para solucionar este problema se puede utilizar el algoritmo de esperanza-maximización (EM). Este algoritmo va entregando iterativamente parámetros que aumentan la probabilidad P$(X|\lambda)$ y terminan convergiendo a un máximo local.

2.4. Deep Neural Network

Las redes neuronales profundas (DNN) son un subcampo dentro del campo de las redes neuronales artificiales (ANN). Las redes neuronales son modelos inspirados en la estructura y el funcionamiento del cerebro. En las ANN las neuronas del cerebro son representadas por nodos, los cuales están interconectados entre si en el sentido de que la salida de un nodo sirve como entrada para otros nodos en la red.

En la figura 2.5 se muestra el perceptrón, que puede ser considerado como una red neuronal de una sola capa. Aquí, los valores de los datos de entrada son ponderados por pesos, luego se suman los resultados y se determina la salida al pasar el resultado por una función de activación.

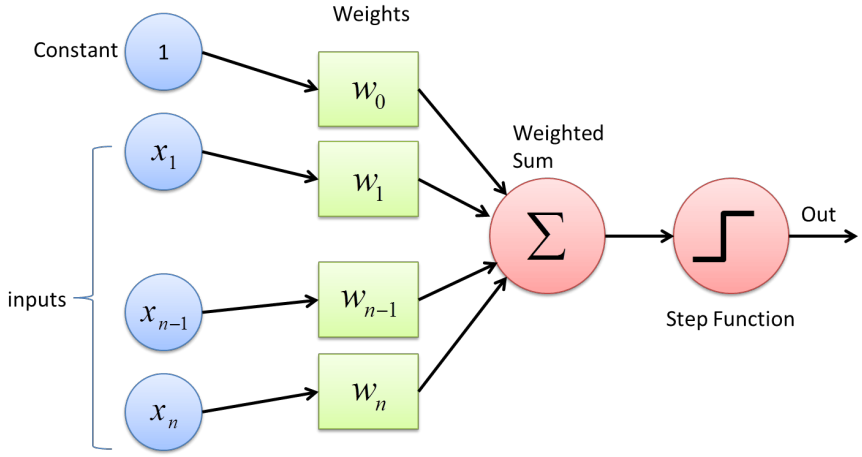

Figura 2.5: Esquema de un perceptrón

El perceptrón puede ser utilizado, por ejemplo, para realizar clasificaciones binarias de objetos a partir de sus características. Aquí, los datos de entrada serían las características de un determinado objeto y se desea que el perceptrón determine si pertenece o no a un grupo (1 o 0 en la salida). Para obtener resultados adecuados, resulta necesario ajustar los pesos del perceptrón, para que su salida concuerde con la etiqueta de este (Este ejemplo representa

un problema de aprendizaje supervisado, ya que se tienen los ejemplos de entrenamiento ya etiquetados).

Claramente las aplicaciones de una red neuronal de una capa (como lo es el perceptrón) no son muy amplias, pero al formar redes neuronales con varias capas es posible aproximar cualquier función. Al tener una red neuronal de varias capas, la salida de los nodos de las capas intermedias se usa para alimentar nodos en la capa siguiente y la capa final entrega la salida de la red. En la figura 2.6 se presenta un esquema de una red DNN.

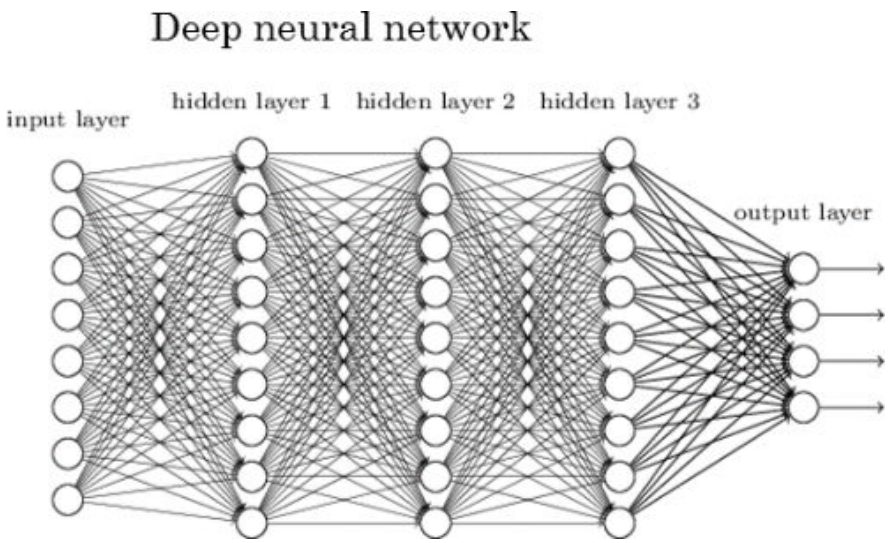

Figura 2.6: Esquema de una red neuronal profunda

Se ha demostrado teóricamente que una red neuronal profunda puede aproximar cualquier función, por compleja que sea, dadas suficientes capas. En Cybenko 1989[16] se prueba que es posible aproximar cualquier función continua (propiedad de universalidad) por medio de perceptrones multicapa, usando funciones de activación sigmoides. En $Sch\ddot{a}fer$ et al. 2006[17] se prueba la propiedad de universalidad de las redes neuronales recurrentes (RNN) y en Zhou 2020[18] se prueba la propiedad para las redes neuronales convolucionales, dadas suficientes capas de profundidad.

Una de las ventajas de las redes neuronales profundas es que, al entrenarlas con mayor cantidad de datos, sus resultados continúan mejorando. Esta característica las diferencia de la mayoría de las otras técnicas de *machine learning* que inicialmente mejoran su rendimiento, pero luego alcanzan una meseta de $performance$. Esta característica es de vital importancia en la situación tecnológica actual, donde los volúmenes de datos son muy grandes y se tiene gran cantidad de información para entrenar modelos. En la figura 2.7 se muestra un gráfico explicativo de este fenómeno, se puede ver como otras técnicas de *machine learning* se estancan y dejan de mejorar, mientras que la DNN sigue mejorando al aumentar la cantidad de datos de entrenamiento.

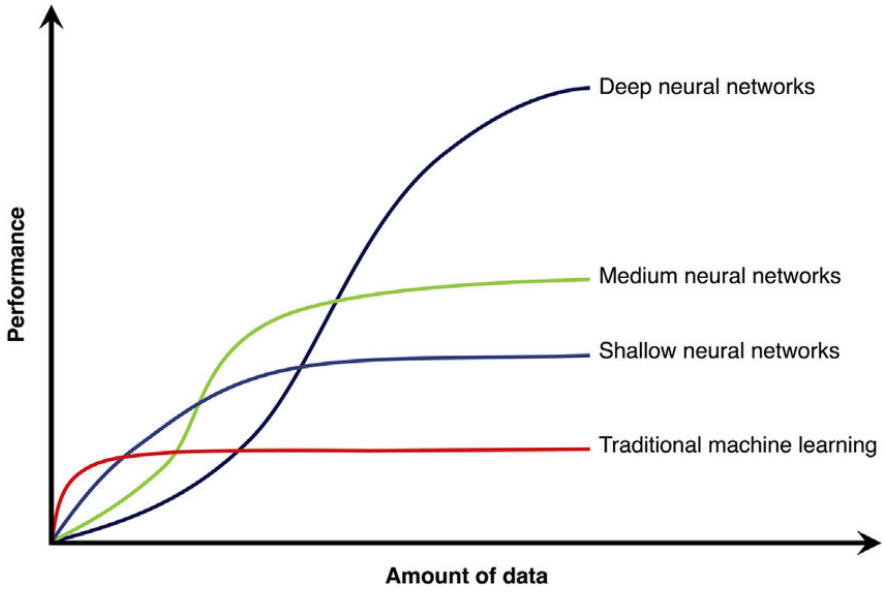

Figura 2.7: Gráfico de *performance* en función de la cantidad de datos usando distintas técnicas

Cabe destacar que la factibilidad de usar redes neuronales profundas se debe al desarrollo tecnológico, que permite realizar la enorme cantidad de computaciones necesarias para ajustar los modelos DNN en tiempos razonables. Sin los avances en la velocidad de los procesadores y las tarjetas gráficas, no sería práctica la utilización de redes profundas.

Los algoritmos de *deep learning* logran encontrar representaciones útiles en la estructura de los datos, generalmente en varios niveles, donde las representaciones de alto nivel se basan en las representaciones más simples. Así, las DNN encuentran jerarquías de representaciones (que pueden ser consideradas como distintos niveles de abstracción) que permiten al modelo mapear funciones muy complejas partiendo simplemente de los datos, sin necesitar representaciones entregadas directamente por los programadores.

Una red neuronal de L capas puede ser definida claramente por medio de expresiones matemáticas. Los pesos de las conexiones entre neuronas se pueden denotar como w^l_{jk}, donde w^l_{jk} corresponde al peso de la conexión que va desde la $k-ésima$ neurona de la $l-1-ésima$ capa de la red neuronal hasta la $j-ésima$ neurona de la $l-ésima$ capa de la red, con $1 \leq l \leq L$. Por ejemplo, en la figura 2.8 se muestra una red neuronal de 3 capas donde se muestra explícitamente el peso de la conexión entre la cuarta neurona de la segunda capa con la segunda neurona de la tercera capa.

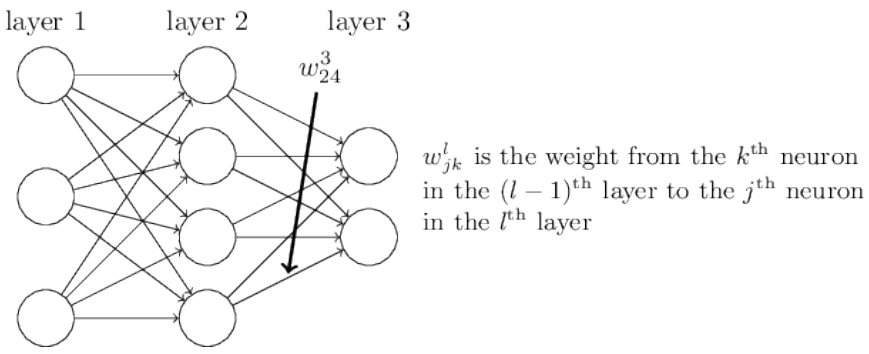

Figura 2.8: Ejemplo de notación de pesos utilizada para describir una red neuronal

De manera similar, se definen los sesgos (*bias*) y las activaciones de cada neurona. Se puede denotar el *bias* de la $j-$ésima neurona de la $l-$ésima capa como b^l_j y su activación como a^l_j. Basados en esta notación, se puede relacionar la activación de la neurona $j-$ésima en la capa $l-$ésima con las activaciones de la capa $l-1-$ésima por medio de la ecuación:

$$a^l_j = \sigma(\sum_k w^l_{jk} a^{l-1}_k + b^l_j)$$

Donde $\sigma(\cdot)$ es la función de activación no lineal utilizada, que puede ser una función sigmoidea, una función $softmax$, la función tangente hiperbólica, etc. Notar que en base a esta ecuación se puede calcular la activación en toda la red si se conocen las activaciones de la capa inicial. Los datos de entrada corresponden a estas "activaciones iniciales", por lo que a partir de datos de entrada se determina la activación de todas las neuronas de la red. Generalmente las activaciones de la capa final se toman como la salida de la red neuronal, así, a partir de datos de entradas, se puede computar directamente la salida de la red. Claramente está definición sigue siendo válida para una red de cualquier número de capas L, por lo que sirve para describir incluso redes neuronal profunda con un valor alto de L.

Para obtener un modelo de red neuronal ajustado adecuadamente a los datos, se requiere sintonizar correctamente los valores de los pesos w^l_{kj} y los bias b^l_j y para esto usualmente se usa el método del gradiente descendiente. Para usar este método de entrenamiento se requiere conocer la salida correcta deseada, por lo que generalmente las redes neuronales son usadas en problemas de aprendizaje supervisado.

2.5. Passive Acoustic Monitoring

El monitoreo acústico pasivo (PAM) es un método ampliamente utilizado para monitorear la presencia de animales marinos (como ballenas, delfines, etc.) en los océanos y mares. Se capta la existencia de estos animales por medio de sus vocalizaciones y permite conocer su presencia tanto temporal como espacial en los ambientes donde se ubican los sensores.

Esta técnica consiste en ubicar en el fondo marino un arreglo de hidrófonos para detectar vibraciones que viajan a través del agua. Específicamente, los hidrófonos detectan cambios en la presión del agua, generadas por las ondas sonoras que viajan a través de esta y generalmente estas señales son almacenadas para ser procesadas posteriormente. Puesto que las vibraciones

de menor frecuencia se absorben en menor medida, estas viajan más lejos, por lo que PAM funciona mejor para detectar sonidos de baja frecuencia. Esto hace que la técnica sea ideal para detectar vocalizaciones de ballenas, delfines y ruidos provenientes del movimiento de las placas tectónicas. Cabe destacar que el agua conduce de mejor manera las vibraciones que el aire, por lo que se pueden detectar sonidos generados a grandes distancias de los hidrófonos.

Los hidrófonos son ubicados generalmente en tripletes, en disposición triangular, lo que permite no solo detectar sonidos, sino también determinar la dirección de la que provienen. En la figura 2.9 se muestra un esquema de cómo se posicionan los hidrófonos en el lecho marino, se puede ver como las señales detectadas son transmitidas vía cable hasta la estación encargada del almacenamiento y procesamiento, ubicada en tierra firme.

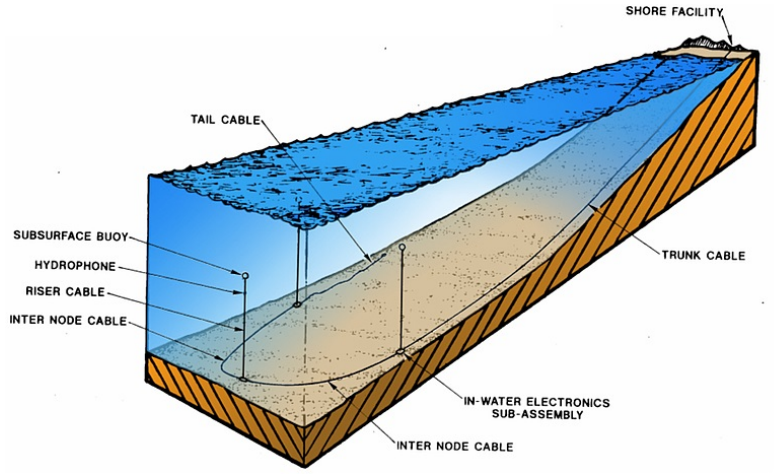

Figura 2.9: Esquema del posicionamiento de un arreglo de hidrófonos en el fondo del mar

En el caso de las ballenas barbadas, por ejemplo, los hidrófonos son capaces de detectar vocalizaciones a decenas de kilómetros desde la fuente. Las vocalizaciones de estas ballenas son repetitivas, de gran amplitud y de baja frecuencia. Estas características permiten que la detección a tan grandes distancias sea posible.

El uso del método PAM para monitorear los océanos presenta grandes ventajas. En particular, permite recolectar información durante todo el año, sin los problemas logísticos y económicos que otros métodos de monitoreo presentan, como los basados en recolección de datos en botes. Esta ventaja resulta esencial en zonas aisladas y remotas donde la recolección de datos por otros métodos resulta infactible.

A la vez, la eficiente transmisión del sonido a través del agua permite monitorear grandes extensiones de agua utilizando una cantidad relativamente baja de estaciones hidroacústicas. Por ejemplo, la Organización del Tratado de Prohibición Completa de los Ensayos Nucleares (CTBTO), encargada de monitorear los océanos para supervisar el cumplimiento del tratado internacional que prohíbe ensayos nucleares, utiliza solo 11 estaciones hidroacústicas formadas por arreglos de hidrófonos para monitorear todos los océanos de la Tierra.

Por otra parte, este método genera bases de datos de gran volumen, que no pueden ser

analizadas manualmente por expertos en su totalidad. Por esta razón, resultan necesarios métodos automáticos de análisis que no requieran demasiada intervención humana.

2.6. Vocalizaciones de Ballenas

Las ballenas azules (*Balaenoptera musculus*) generan distintos tipos de vocalizaciones, entre las que se tienen: frases de canto *Southeast Pacific 1* (*SEP1*), *Southeast Pacific 2* (*SEP2*) y vocalizaciones no-canto *D-calls*.

El canto SEP1 de ballena azul consiste en frases de 3 unidades tonales (denotadas como A, B, C y generadas en ese orden) de baja frecuencia. La unidad A es un sonido tonal con frecuencia *peak* alrededor de $20Hz$ y $10-15s$ de duración. La unidad B del canto corresponde a un sonido tonal de frecuencia *peak* alrededor de $25Hz$ y 10s de duración. La unidad C es un sonido tonal de 9s de duración y frecuencia *peak* de $25Hz$ aproximadamente (Cummings and Thompson 1971 [19]; Buchan et al. 2014 [20]). En la figura 2.10 se muestra un ejemplo de este canto, en la imagen se presenta un espectrograma donde se distinguen las tres unidades de esta vocalización.

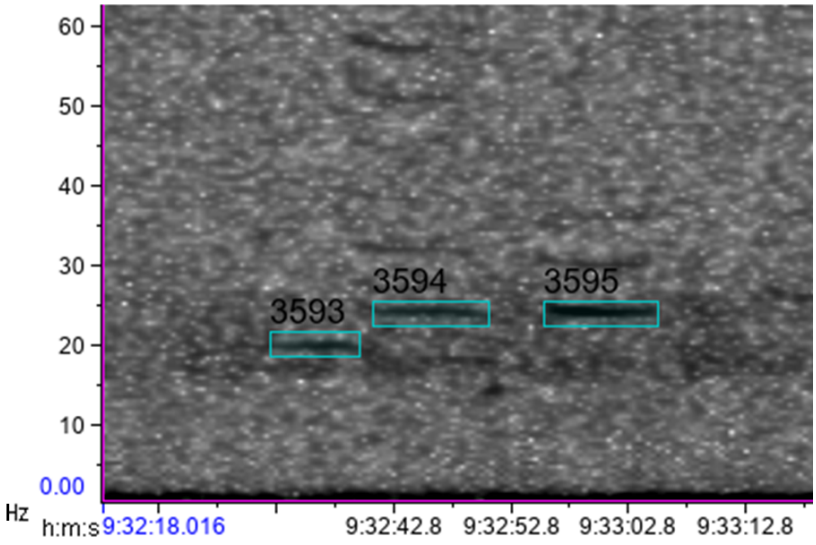

Figura 2.10: Ejemplo de vocalización SEP1 de ballena azul

El canto SEP2 consiste en frases de 4 unidades tonales de baja frecuencia, denotadas A, B, C, D. La unidad A es un sonido tonal con frecuencia *peak* alrededor de los 25Hz y de duración entre $12s$ y $15s$. La unidad B es un sonido pulsado de frecuencia *peak* de $95Hz$ aproximadamente y alrededor de $12s$ de duración. La unidad C es un sonido tonal de duración aproximada $4s$ y frecuencia alrededor de 25Hz. La unidad D es un sonido tonal de entre $4s$ y $7s$ de duración. Esta unidad inicia en $25Hz$ durante aproximadamente $4s$ y termina con una bajada de frecuencia hasta alrededor de los $17Hz$ (Buchan et al. 2014[20]; Buchan et al. 2015 [4]). En la figura 2.11 se muestra un ejemplo de canto $SEP2$, en la subfigura superior se muestra la señal en el dominio temporal y en la subfigura inferior se muestra el espectrograma de la

señal. Se pueden apreciar las cuatro unidades de la vocalización en el domino de la frecuencia.

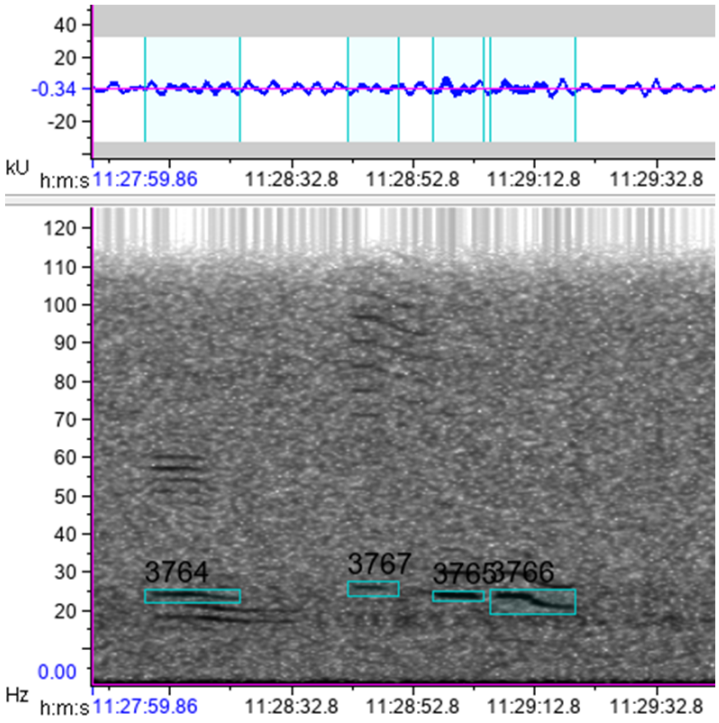

Figura 2.11: Ejemplo de vocalización SEP2 de ballena azul

Las vocalizaciones no canto $D - calls$ se caracterizan por ser bajadas de frecuencia (*downsweeps*) entre $100Hz$ y $30Hz$. Los rangos de frecuencia y duración de la vocalización son variables. Se distinguen de los *downsweeps* de ballena fin por tener mayor duración, generalmente en el rango de $1s$ a $5s$ (Saddler et al. 2017[21]; Schall et al. 2019[22]). En la figura 2.12 se presenta un ejemplo de $D-call$, en la subfigura superior se muestra la señal en el dominio temporal y en la subfigura inferior se muestra el espectrograma. Se puede apreciar la caída en frecuencia (*downsweeps*) de la vocalización, dentro del rango de frecuencias mencionado.

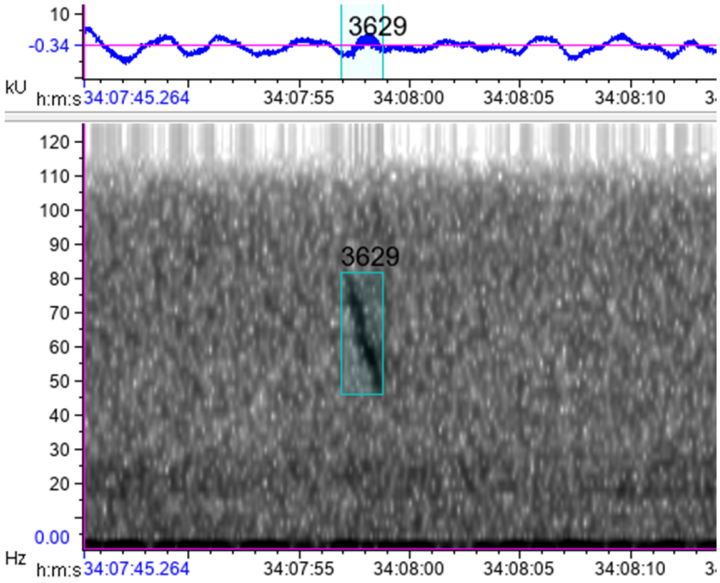

Figura 2.12: Ejemplo de $D - call$ de ballena azul

La subespecie de ballena azul del océano Antártico (Balaenoptera musculus intermedia) produce un tipo característico de vocalización conocida como $z - call$. El sonido está compuesto de tres unidades, denominadas A, B y C. La primera unidad tiene una frecuencia en el rango de frecuencias $28Hz - 29Hz$ y tiene una duración de entre $8s$ y $12s$. La unidad B es un corto *downsweep* de $2s$ de duración y frecuencia entre $28Hz$ y $20Hz$, que conecta la primera unidad con la tercera. La unidad C es un tono ligeramente modulado de duración en el rango $8s - 12s$ y frecuencia dentro del rango $20Hz - 18Hz$ (Stafford et al. 2004[23]; Miller et al. 2007[24]). En la figura 2.13 se presenta un ejemplo con tres $z - call$ distintas, en la subfigura superior se muestra la señal en el dominio temporal y en la subfigura inferior se muestra el espectrograma de la señal. Se puede apreciar las tres unidades de este tipo de vocalización.

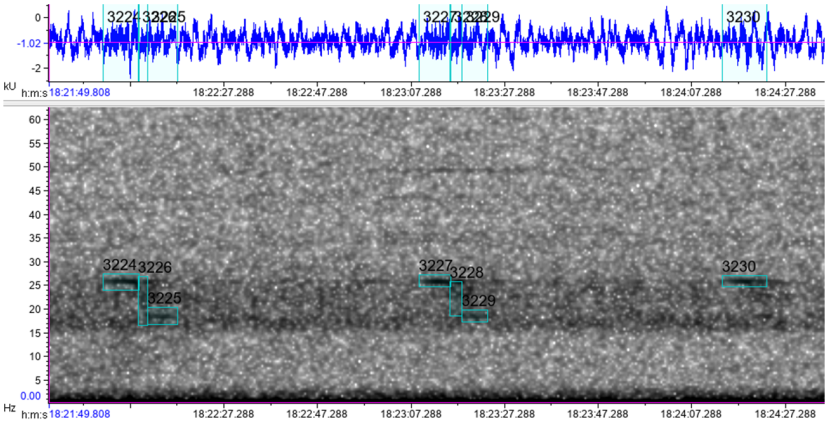

Figura 2.13: Ejemplo de $z - call$ de ballena azul antártica

Las ballenas fin (*Balaenoptera physalus*) producen distintas vocalizaciones, entre las que se tienen las frases de canto *FW.S* (*Fin whale song*) y tres tipos diferentes de *downsweep* *FW.D* (*Fin whale downsweep*).

Los cantos *FW.S* son vocalización producidas sólo por ballenas fin machos. En la literatura esta vocalización es conocida como "canto de 20Hz" y consiste en una vocalización pulsada, con un componente de baja frecuencia tipo *downsweep* en el rango $30Hz - 15Hz$ y un componente de alta frecuencia en el rango $80 - 90Hz$ (Watkins et al. 1987[25]; Charif et al. 2001[26]; Širović et al. 2007 [27]; Gedamke 2009[28]). En la figura 2.14 se presenta un ejemplo de canto *FW.S*, donde se muestra el espectrograma de la señal original con las características mencionadas.

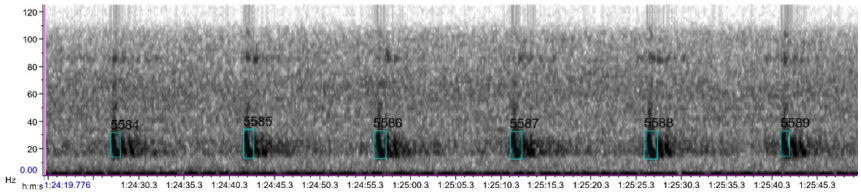

Figura 2.14: Ejemplo de canto *FW.S* de ballena fin

El primer tipo de vocalización *dowsweep* de ballena fin posee ancho de banda variable. La frecuencia varía entre $80Hz$ y $30Hz$, con una duración en el rango $0.7s - 0.9s$. Al igual que muchos *downsweeps*, cuando son fuertes presentan de 1 a 2 armónicos (Ou et al. 2015[29]; Watkins et al. 1981[30]). En la figura 2.15 se presenta un ejemplo de esta vocalización, en la subfigura superior se muestra la señal en el dominio del tiempo y en la subfigura inferior se muestra el espectrograma de la señal. Se puede apreciar la caída en frecuencia (*downsweeps*) de la vocalización, conforme al rango de frecuencias y de duración descritos.

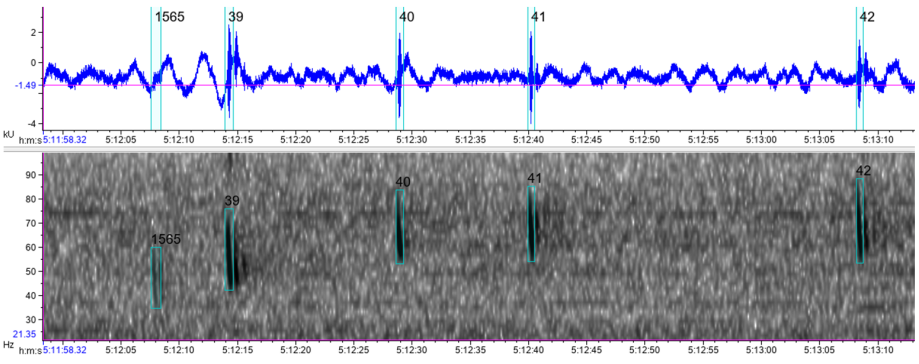

Figura 2.15: Ejemplo del primer tipo de vocalización *dowsweep* de ballena fin

El segundo tipo de vocalización *dowsweep* de ballena fin también posee ancho de banda variable, esta vez en el rango de frecuencias entre $40Hz$ y $20Hz$. Tiene una duración similar

al primer tipo de *dowsweep*, en el rango $0.7s - 0.9s$. Cuando son fuertes también presentan de 1 a 2 armónicos (Watkins et al. 1981[30]). En la figura 2.16 se presenta un ejemplo de esta vocalización, en la subfigura superior se muestra la señal en el dominio temporal y en la subfigura inferior se muestra el espectrograma de la señal. Se puede apreciar la caída en frecuencia (*downsweeps*) de la vocalización, según el rango de frecuencias y de duración descritos.

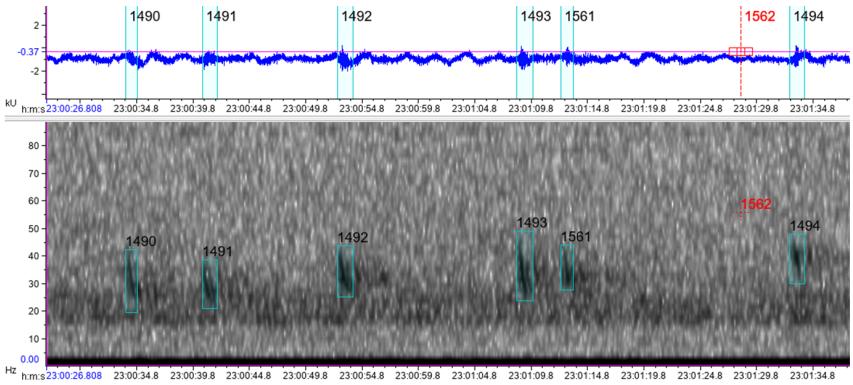

Figura 2.16: Ejemplo del segundo tipo de vocalización *dowsweep* de ballena
fin

El tercer tipo de vocalización *dowsweep* también posee ancho de banda variable. El rango de frecuencias va desde los $30Hz$ hasta los $15Hz$ y la duración de la vocalización es variable. Los intervalos parecen regulares en algunas partes de la frecuencia, e irregulares en otras. Cuando son fuertes también presentan de 1 a 2 armónicos. En la figura 2.17 se presenta un ejemplo de esta vocalización, en la subfigura superior se muestra la señal en el dominio del tiempo y en la subfigura inferior se muestra el espectrograma de la señal. Se puede apreciar que la caída en frecuencia (*downsweeps*) de la vocalización concuerda con el rango de frecuencias y de duración descritos.

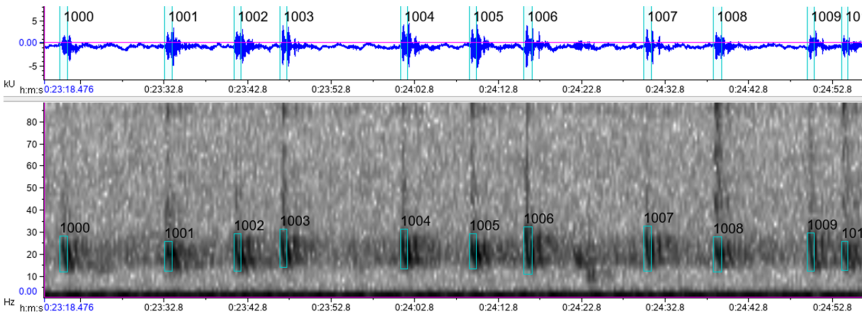

Figura 2.17: Ejemplo del tercer tipo de vocalización *dowsweep* de ballena
fin

Las ballenas sei (*Balaenoptera borealis*) producen distintas vocalizaciones, clasificadas en vocalizaciones tipo *upsweep SW.U* (*Sei whale upsweep*) y vocalizaciones tipo *downsweep SW.D* (*Sei whale downsweep*).

Las vocalizaciones tipo *upsweeps* de las ballenas sei se caracterizan por ser subidas de frecuencia de $30Hz$ de ancho de banda, con frecuencia inferior a $100Hz$ y usualmente entre $40Hz$ y $70Hz$. La duración de estas vocalizaciones es de $1.2s$ aproximadamente y se reconocen por venir en conjuntos (Calderan et al. 2014[31]). En la figura 2.18 se muestra el espectrograma de una señal con vocalizaciones *upsweep*, se puede apreciar como la frecuencia de las vocalizaciones va subiendo y está dentro de los rangos de frecuencia descritos.

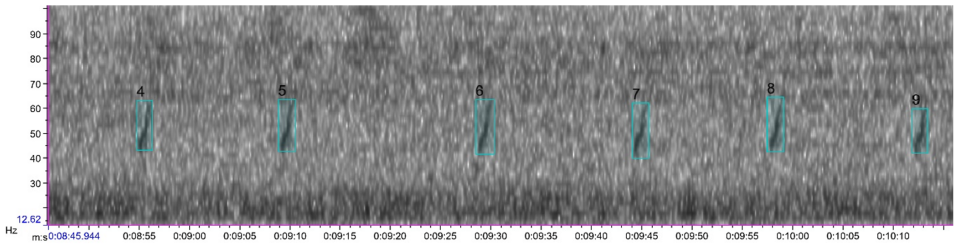

Figura 2.18: Ejemplo de vocalización tipo *upsweep* de ballena sei

Las vocalizaciones tipo *downsweeps* de las ballenas sei se caracterizan por ser bajadas de frecuencia que pueden estar en el rango de frecuencias $90Hz - 35Hz$ o en el rango $80Hz - 35Hz$. La duración de estas vocalizaciones es de $1.6s$ aproximadamente y se reconocen por venir en conjuntos de dos o de tres vocalizaciones (Español-Jimenez et al. 2019[32]; Ou et al. 2015[29]). En la figura 2.18 se muestra el espectrograma de una señal con vocalizaciones tipo *dowsweep*, se puede ver como la frecuencia de las vocalizaciones disminuye al avanzar el tiempo.

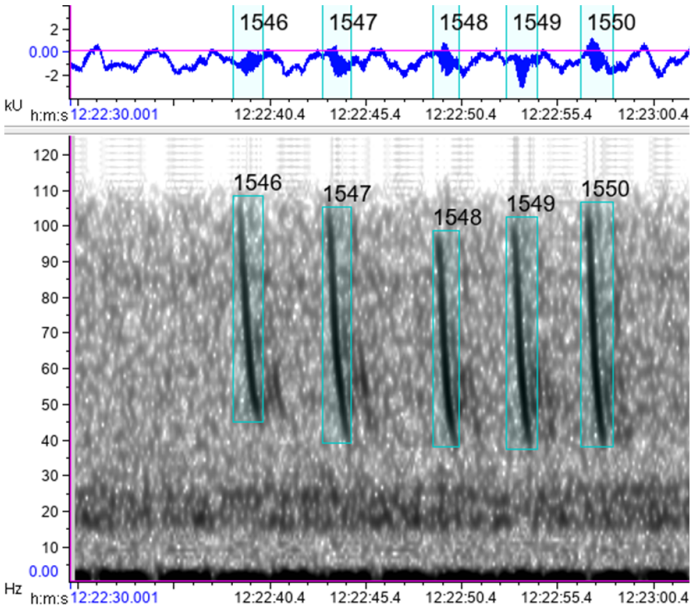

Figura 2.19: Ejemplo de vocalización tipo *downsweep* de ballena sei

2.6.1. Otros

Los sismos presentan tres diferentes tipos de ondas, denominadas ondas S, ondas P y ondas T. Por transmisión marítima generalmente se observan ondas T, seguidas de ondas P. Las señales de sismos son de baja frecuencia con textura "rasposa" en el espectrograma y con la mayor cantidad de energía concentrada en el rango de frecuencias $1Hz - 50Hz$. La forma de onda en el espectrograma se observa como rápido aumento de amplitud y luego un descenso (De Angelis et al. 2007[33]; Fox et al. 2001[34]; Yun et al. 2009[35]; Caplan-Auerbach et al. 2014[36]; Dziak et al. 1997[37]). En la figura 2.20 se presenta un ejemplo de esta señal, en la subfigura superior se muestra la señal en el dominio del tiempo y en la subfigura inferior se muestra su espectrograma. Se puede apreciar la baja frecuencia de la señal y su repentino comienzo.

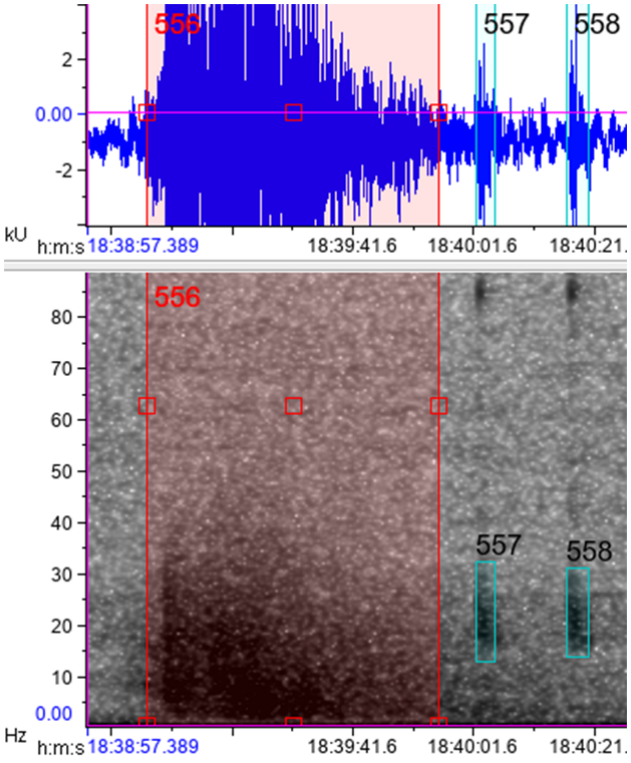

Figura 2.20: Ejemplo de señal correspondiente a un sismo

Al avanzar a través del agua, los barcos generan vibraciones con características distintivas de este tipo de evento. Las señales alcanzan altos decibeles de amplitud, en especial en el rango de frecuencias $1Hz - 150Hz$, pero pueden alcanzar frecuencias medianas en el espectrograma (1000Hz) (producidas cuando el barco está transitando cerca del sensor o cuando aumenta su velocidad). La duración de la señal es variable y depende de la velocidad del barco, aunque generalmente supera los 5 minutos (McKenna et al. 2012[38]; Lourens 1988[39]; Simard et al. 2014[40]; Stanley et al. 2017[41]). En la figura 2.21 se presenta un ejemplo de esta señal producida por un barco (recuadro rojo), en la subfigura superior se muestra la señal en el dominio temporal y en la subfigura inferior se muestra su espectrograma. Se puede

apreciar la gran amplitud de la señal y su larga duración.

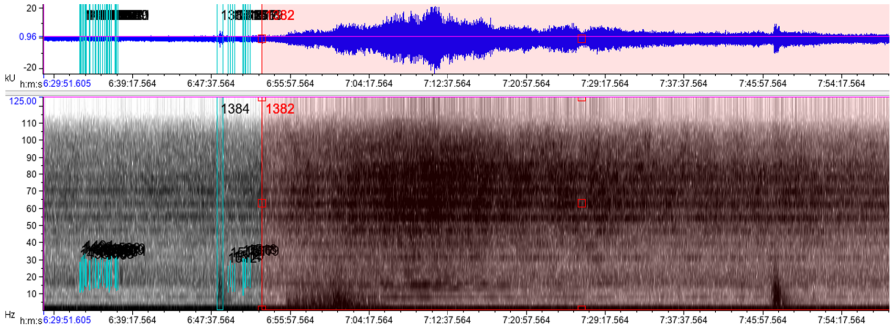

Figura 2.21: Ejemplo de señal correspondiente a un barco

Existen sonidos cuyo origen no se puede determinar con certeza y se clasifican como indefinidos. Son sonidos transitorios, de pulso de banda ancha, mayor a los $30Hz$ y de corta duración (generalmente en el rango $3s - 5s$). En la figura 2.22 se presenta un ejemplo de una señal clasificada como indefinida, en la subfigura superior se muestra la señal en el dominio del tiempo y en la subfigura inferior se muestra su espectrograma. Se puede observar su gran ancho de banda y su corta duración.

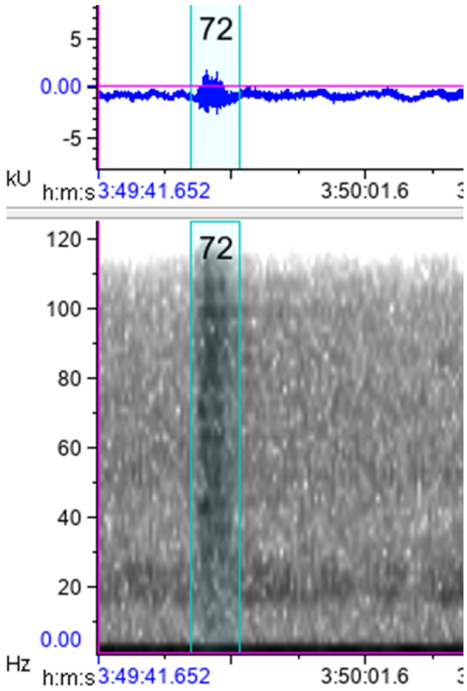

Figura 2.22: Ejemplo de señal clasificada como indefinida

2.7. Software Kaldi

Kaldi es un *toolkit* diseñado para proyectos relacionados con el reconocimiento del habla. Este *software* de código abierto fue creado con la intención de ser útil para investigadores y profesionales del área de reconocimiento de voz.

El objetivo de Kaldi es entregar códigos flexibles que sean fáciles de modificar y de extender. La idea es proveer algoritmos generales en vez de específicos, de manera que puedan ser usados en distintas situaciones y casos. El código está diseñado para ser compresible sin mucho esfuerzo, en el sentido de que cada parte individual es fácil de entender.

Kaldi fue lanzado bajo la licencia *Apache v2.0* que es altamente no restrictiva, por lo que puede ser utilizada por una amplia comunidad de usuarios.

Figura 2.23: Logo de software Kaldi

En la actualidad, Kaldi tiene código para la mayoría de las técnicas estándar, como transformaciones lineales, *Deep Neural Network* (DNN), GMM (*Gaussian Mixture Models*), *Maximum Mutual Information* (MMI) y *Boosted MMI*. A la vez, Kaldi es capaz de generar vectores de características con técnicas como *Mel-Frequency Cepstrum* (MFCC), bancos de filtros, *Maximum Likelihood Linear Regression* (MLLR), entre otras. Todas estas herramientas lo convierten en un software muy útil para problemas de reconocimiento automático de voz (ASR)[42].

2.8. Espectrograma

Un espectrograma es una representación del espectro de frecuencias de una señal, que muestra la energía de distintas frecuencias a medida que avanza el tiempo. En general, un espectrograma tiene tres dimensiones: el tiempo, la frecuencia y la energía de cada frecuencia. Se suele mostrar la frecuencia en el eje de las ordenadas (eje Y), el tiempo en el eje de las abscisas (eje X) y la energía es representada en escala de colores. En la figura 2.24 se muestra un espectrograma en la subfigura inferior y la señal original en el dominio del tiempo en la subfigura superior. En el espectrograma de ejemplo, el color blanco representa la mayor magnitud de energía, y va bajando, pasando por el amarillo, luego el rojo, luego el azul, hasta llegar al negro que representa ausencia de energía. Se puede ver en la figura como los eventos en la señal original son descompuestos en frecuencias en el espectrograma y la gran mayoría de la energía aparece en las frecuencias bajas del espectro considerado.

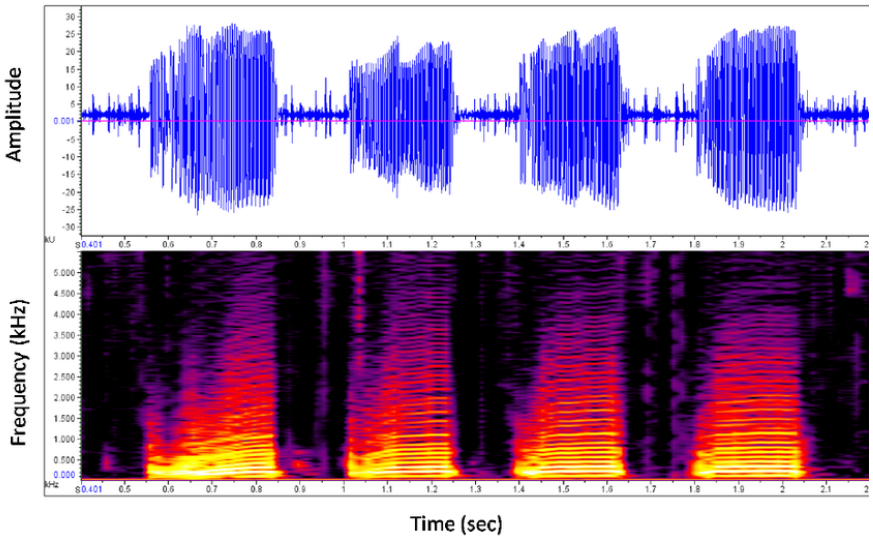

Figura 2.24: Espectrograma de ejemplo

Para obtener un espectrograma a partir de una señal discreta, se suele utilizar la transformada rápida de Fourier. La metodología consiste en tomar una ventana temporal de la señal, calcular sus componentes en el dominio de la frecuencia con FFT y mostrar esta descomposición como una línea vertical el espectrograma. Luego se toma una ventana adyacente, se aplica FFT y se muestra esta descomposición como la línea vertical siguiente en el espectrograma, y así hasta tomar toda la señal y obtener un espectrograma completo. Cabe destacar que estas ventanas adyacentes usualmente están solapadas en alguna medida.

2.9. FFT

La transformada rápida de Fourier (*Fast Fourier Transform*) es una manera eficiente de computar la transformada discreta de Fourier (DFT). Cuando se calcula la transformada de Fourier de una señal, se convierte la señal desde su dominio original al dominio de la frecuencia, lo que tiene numerosas aplicaciones. La transformada de Fourier permite conocer las componentes de las distintas frecuencias que conforman la señal original.

En la actualidad se dispone de gran cantidad de señales discretas capturadas por micrófonos, cámaras y otros sensores, y resulta útil en muchas aplicaciones calcular la transformada discreta de Fourier. La DFT permite obtener la representación en el espacio de frecuencia de la señal a partir de muestras discretas de esta. La DFT es calculada a partir de la fórmula:

$$X_k = \sum_{n=0}^{N-1} x_j e^{-i2\pi kj/N} \qquad k = 0, \ldots, N-1,$$

Donde N es el número de muestras disponibles de la señal, X_k es el peso de la frecuencia $k-$ésima y x_j es la muestra $j-$ésima de la señal.

En la práctica, esta fórmula presenta problemas al tratar de computarla. La evaluación de esta fórmula requiere operaciones de orden $O(N^2)$, lo cual es demasiado cuando se tienen señales con gran número de muestras. La solución a este problema práctico es la trasformada

rápida de Fourier, que reduce el orden de las operaciones a realizar a $O(NlogN)$.

El algoritmo FFT se basa en una inteligente reorganización de términos, separando los componentes pares y los impares. Esto permite calcular la FFT de una señal de largo 2^n fácilmente a partir de la FFT de una señal de largo 2^{n-1}, lo que reduce en gran medida la cantidad de operaciones a realizar[43].

Para ejemplificar la aplicación de la transformada discreta de Fourier se presenta la imagen de la figura 2.25. En la subfigura superior se tiene una señal compuesta de la suma de cinco cosenos según la formula $\sum_{n=1}^{5} n \cdot cos(nwt)$, con $w = 20\pi$ y en la subfigura inferior se muestra la misma señal en el domino de las frecuencias, luego de aplicar la DFT.

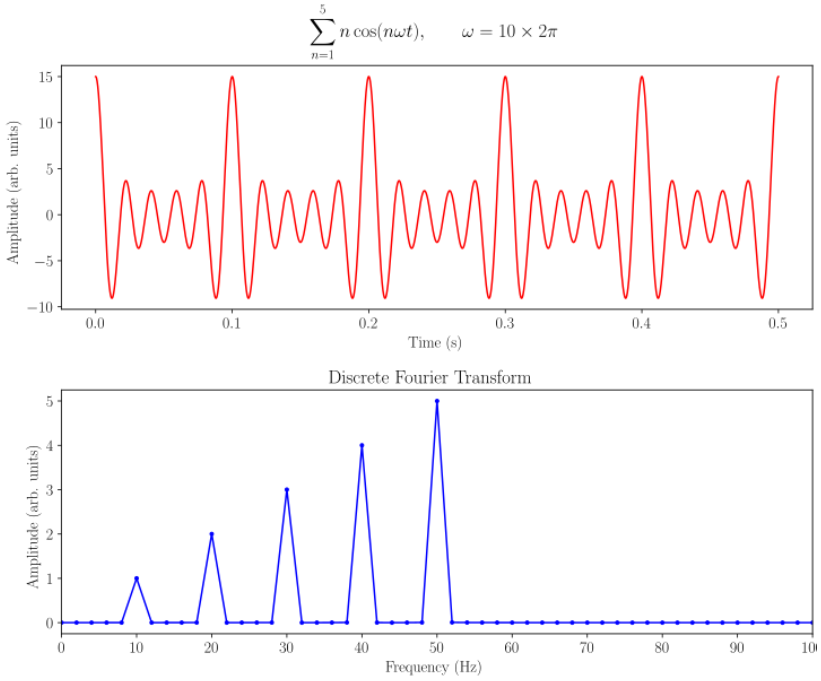

Figura 2.25: Ejemplo de una FFT de una suma de cosenos

Se puede ver como la representación en el dominio de la frecuencia permite ver claramente que la señal está compuesta de 5 frecuencias principales, información difícil de captar observando sólo la señal en el dominio temporal. Más aún, la DFT permite ver la importancia relativa entre cada una de las frecuencias que componen la señal. Este ejemplo muestra la gran cantidad de información relevante que se puede obtener de una señal al utilizar la transformada discreta de Fourier.

2.10. Terminología procesamiento de la voz

2.10.1. Utterance

Una *utterance* es una parte del habla que parte en una pausa clara y termina en una pausa clara. La *utterance* es considerada la unidad más pequeña del habla. Una *utterance* puede ser una palabra dicha, por ejemplo "*Hola*", o puede ser una oración, por ejemplo "*¿Como has*

estado?" dicho de corrido. La definición no se limita solo al habla humana y también abarca vocalizaciones emitidas por otros animales. Los sonidos de cualquier animal pueden contener unidades correspondientes a *utterances*, siempre y cuando cumplan con estar claramente delimitados por pausas claras.

2.10.2. Evento

Un evento corresponde a un objeto que aparece en la señal considerada. Por ejemplo, si se dispone de grabaciones acústicas de un partido de fútbol, un grito de gol correspondería a un evento particular dentro de la grabación. Cabe destacar que la definición de los eventos depende completamente del problema particular que se está abordando y se definen en concordancia con los objetivos del trabajo. Generalmente a cada evento definido se le asocia una etiqueta característica, que luego permite trabajar las señales (con sus anotaciones de eventos asociadas) con métodos computacionales.

En muchas aplicaciones de procesamiento de voz, se definen los eventos presentes en las grabaciones como las *utterance* de las vocalizaciones que aparecen. Esta selección resulta útil puesto que las pausas entre cada *utterance* usualmente son fáciles de captar por los modelos computacionales y los eventos quedan claramente delimitados.

2.10.3. Frame

Un $frame$ corresponde a una subsección de una grabación o señal. Por ejemplo, se puede tener una señal de $1000s$ y dividirla en 10 $frames$ de $100s$ cada uno (considerando $frames$ sin traslape). La utilidad de dividir la señal en $frames$ es que se puede describir como una secuencia de $frames$ y luego se puede trabajar directamente con esta nueva representación. Una aplicación muy común es utilizar la división en $frames$ para obtener vectores de $features$. Así, la división de la señal en $frames$ permite trabajar más fácilmente con la señal para generar modelos computacionales.

2.10.4. Transcripción

La transcripción de una señal es una secuencia de etiquetas correspondiente a la secuencia de eventos ocurridos en la señal. Por ejemplo, si una señal corresponde a la oración "*Bien, gracias*" la transcripción de la señal sería: *bien - silencio - gracias*. Suponiendo que se considera la palabra "*Bien*" como un solo evento, la pausa como otro evento y la palabra "*gracias*" como un evento individual.

2.10.5. Alineamiento

La definición de alineamiento está íntimamente ligada a la de $frame$. En procesamiento de voz, cuando se divide una señal en una secuencia de $frames$, generalmente resulta necesario asignar una etiqueta a cada uno de los $frames$. Así un alineamiento es una secuencia de etiquetas donde cada etiqueta está asociada a un $frame$ de la grabación. Por ejemplo, si se tiene una grabación dividida en 5 $frames$ y se tienen las posibles etiquetas A y B, un posible alineamiento es $A-A-A-A-A$, otro posible alineamiento es $A-B-B-A-A$, etc.

Resulta importante mencionar que en procesamiento de voz generalmente se dispone de la transcripción correcta de la señal (para el entrenamiento), por lo que obtener un alineamiento corresponde a determinar la distribución de las etiquetas que cumplan con la transcripción. Por ejemplo, si uno sabe a priori sabe que la transcripción es $A - B$, posibles alineamientos correctos son $A - A - A - B - B$, $A - B - B - B - B$, etc. En este caso, por ejemplo, $A - A - A - B - A$ no sería un alineamiento correcto.

2.11. Ventana de Hamming

Una ventana, en el contexto matemático, corresponde a una función que es cero fuera de cierto intervalo y no nula dentro del intervalo de interés. Generalmente tiene su máximo en el centro del intervalo, siendo simétrica con relación a este centro. Usualmente esta función es aplicada sobre otra función o señal de manera que el resultado sea cero fuera del intervalo de interés.

La ventana de Hamming es una ventana que tiene la particularidad de que reduce el fenómeno de *spectral leakage* al calcular la FFT de una señal. Esta ventana es un caso especial de la ventana de cosenos que sigue la ecuación [44]:

$$w[n] = \frac{25}{46} - \frac{21}{46} \cdot \cos\left(\frac{2\pi n}{N}\right), \quad 0 \leq n \leq N$$

Donde N es el largo de la ventana considerada.

En la subfigura izquierda de la figura 2.26 se muestra la ventana de Hamming, simétrica con respecto a su centro y que se atenúa a medida que se aleja de este. En la subfigura derecha se muestra la transformada de Fourier de esta función.

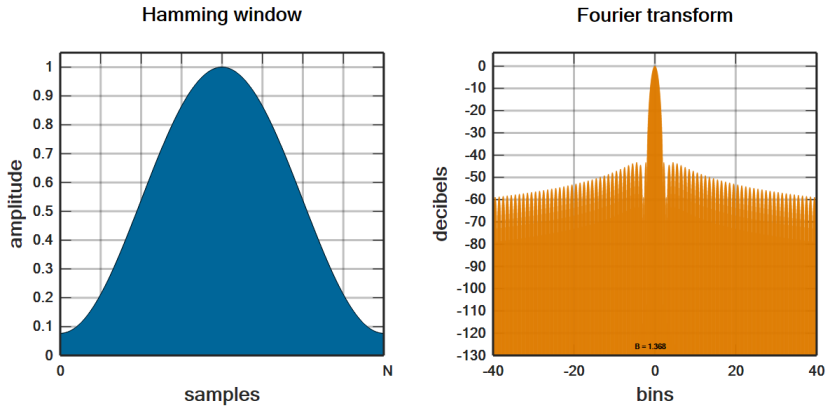

Figura 2.26: Ventana de Hamming y su transformada de Fourier

Como se mencionó, la utilidad de esta ventana aparece al calcular la transformada rápida de Fourier de una señal. En general, cuando se desea calcular la FFT de una señal, implícitamente se está aplicando a una señal que se repite infinitamente y esto genera problemas al trabajar con señales finitas. Un ejemplo claro es tomar una señal simple sinusoidal finita, donde el inicio de la señal no calza con el final. Esta incongruencia será considerada como una discontinuidad de la señal al calcular la FFT, lo que entregará señales de alta frecuencia en el dominio del tiempo. Claramente esto es un error, ya que la señal está compuesta de una

sola frecuencia. Este problema se extrapola a una señal más compleja, ya que está compuesta de varias sinusoidales de distintas frecuencias, por lo que este fenómeno, denominado *spectral leakage* aparecerá de todas maneras[45].

En la figura 2.27 se muestra como aparece el fenómeno al considerar una simple función seno. En azul se muestra un intervalo del seno en que se toma un número exacto de ciclos y el inicio calza con el final de la señal, en cambio, la señal roja corresponde a un intervalo de la misma longitud, pero donde se toma un número no entero de ciclos. Al calcular la DFT de la señal azul, se obtiene solo un componente de frecuencia y en la frecuencia correcta, en cambio, la DFT de la señal roja entrega varias componentes de frecuencias que la señal original no tiene realmente, debido al *spectral leakage* ocurrido.

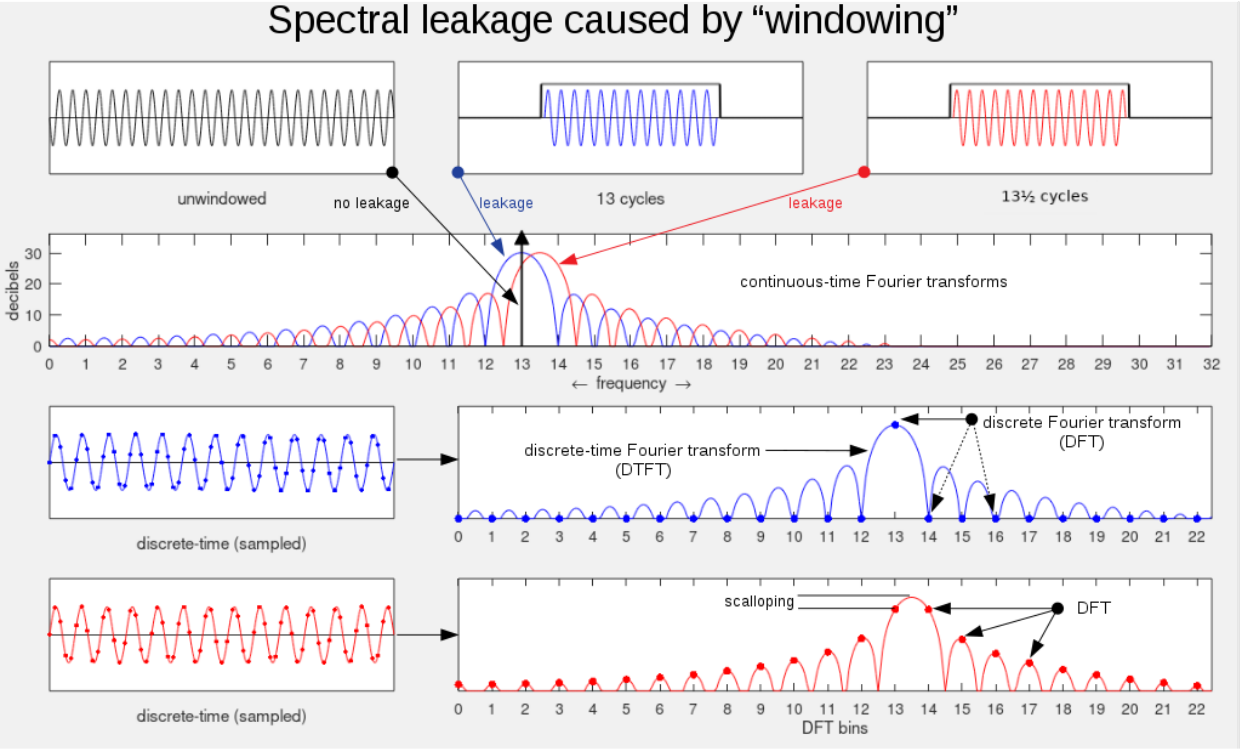

Figura 2.27: Explicación del fenómeno de *spectral leakage*

Para solucionar este problema, se puede multiplicar primero la señal por una ventana de Hamming, con lo que el final y el inicio de la señal calzan. Así, la discontinuidad y el fenómeno de *spectral leakage* disminuyen.

2.12. Autoencoder

Los *autoencoders* son una técnica de aprendizaje no supervisado que utiliza redes neuronales para aprender representaciones eficientes de los datos de entrada. En términos simples, un *autoencoder* es una red neuronal que aprende a entregar como salida el mismo vector de entrada aplicado a la red. La red tiene capas ocultas que pueden ser usadas como nuevas representaciones de los datos de entrada, puesto que es posible reconstruir los datos de entrada a partir de estas capas[46].

En la figura 2.28 se muestra el esquema de un *autoencoder* simple, se tiene una capa inicial asociada a los datos de entrada denotada por $\mathbf{X}$, luego una capa oculta $\mathbf{A}$ y finalmente la capa de salida $\hat{\mathbf{X}}$. La capa de entrada y la de salida tienen la misma dimensión puesto que el objetivo del *autoencoder* es recuperar como salida el mismo vector de entrada. Notar que si el *autoencoder* está correctamente entrenado, la capa oculta posee toda la información necesaria para recuperar el vector de entrada, por lo que puede ser utilizado como una nueva representación del vector de entrada. Usualmente se aprovecha esta característica para obtener nuevas representaciones más compactas de los datos, colocando una capa oculta (denominada también capa de codificación) de menor largo que la capa de entrada, tal como se muestra en la figura 2.28.

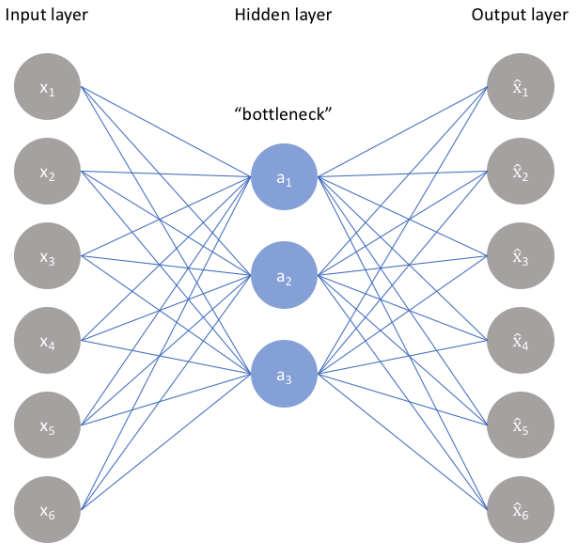

Figura 2.28: Esquema de una red neuronal autoencoder

En términos matemáticos el *autoencoder* se describe de forma análoga a una red neuronal. En el caso de ejemplo de la figura 2.28 la capa oculta cumple con la ecuación:

$$\mathbf{A} = \sigma(W\mathbf{X} + b)$$

Donde $\sigma(\cdot)$ es la función de activación, W son los pesos para pasar de la capa de entrada a la capa oculta y b son los *bias* entre la capa de entrada y la capa oculta. Para la capa de salida se tiene la relación:

$$\hat{\mathbf{X}} = \sigma(\hat{W}\mathbf{A} + \hat{b})$$

Donde σ es la función de activación, $\hat{W}$ son los pesos para pasar de la capa oculta a la capa de salida y $\hat{b}$ son los bias entre la capa oculta y la capa de salida. El objetivo del entrenamiento del *autoencoder* en términos matemáticos corresponde a minimizar el error:

$$C(\mathbf{X}, \hat{\mathbf{X}})$$

Donde C es la función de costo evaluada entre el vector de entrada y el vector de salida de la red. Generalmente se usa la función de mínimos cuadrados para minimizar la distancia

entre $\hat{\mathbf{X}}$ y $\mathbf{X}$, en cuyo caso se tiene $C = ||\mathbf{X} - \hat{\mathbf{X}}||^2$. De esta manera, el entrenamiento se enfoca en entregar una salida $\hat{\mathbf{X}}$ similar al vector de entrada $\mathbf{X}$.

Capítulo 3

Datos Disponibles

Los datos utilizados en este trabajo provienen de arreglos de hidrófonos de una estación hidroacústica ubicada en las costas de la isla de Juan Fernández, Chile. Las grabaciones son capturadas por medio de la técnica de Monitoreo Acústico Pasivo (PAM). La estación hidroacústica está conformada por 2 arreglos de hidrófonos, uno ubicado al norte de la isla (compuesto de 3 hidrófonos denominados $1N$, $2N$ y $3N$) y otro arreglo ubicado al sur de la isla (compuesto de 3 hidrófonos denominados $1S$, $2S$ y $3S$). Este posicionamiento de los hidrófonos permite evitar el efecto sombra que la isla genera al bloquear físicamente los sonidos provenientes de la dirección opuesta a un arreglo de hidrófonos.

La estación hidroacústica y los datos recolectados pertenecen a la Organización del Tratado de Prohibición Completa de los Ensayos Nucleares (CTBTO). Los datos son suministrados al proyecto por la Dra. Susannah Buchan quien tiene acceso a los datos gracias a un convenio de investigación entre la Universidad de Concepción y la Universidad de Washington EEUU.

Los datos disponibles son grabaciones continuas capturadas con una tasa de muestreo de 250 Hz. Los datos originalmente llegan en formato *.w* y son convertidos a formato *.wav* para ser usados posteriormente. En el trabajo actual se utilizan grabaciones provenientes del hidrófono $1N$ del arreglo ubicado al norte de la isla.

Las grabaciones disponibles son filtradas por analistas entrenados de la Universidad de Concepción, en busca de señales de interés con presencia de eventos relevantes. Posteriormente, las grabaciones seleccionadas son etiquetadas manualmente por los analistas utilizando el *software Raven Pro*. Se usa una ventana de Hamming con 80 % de *overlap* y 5 minutos (30 minutos en el caso especial de los eventos de barco) en el eje del tiempo para la visualización del espectrograma. El etiquetado contiene el tiempo de inicio y de término de cierto evento dentro de la grabación, su etiqueta característica y datos como la frecuencia de *peak*, entre otros. En el trabajo se utiliza solo el tipo de evento (etiqueta) y los tiempos de inicio y final de este.

Gracias al convenio, se dispone la base de datos completa de los años 2016, 2017 y 2018 (hasta agosto) de las grabaciones de la estación HA03 del CTBTO en Juan Fernández. De cada año, se seleccionaron los meses de invierno (junio, julio, agosto) para la visualización, puesto que los análisis previos indican mayor presencia de señales de interés en estos meses. Los datos disponibles, compuestos de las grabaciones con sus anotaciones respectivas, corres-

ponden a 4 entregas: la primera compuesta de 34 horas de audio del mes de junio de 2016, la segunda compuesta de 48 horas de datos de agosto de 2016, la tercera compuesta de 114 horas correspondientes a los meses junio, julio y agosto de 2016, y finalmente una entrega compuesta de 146 horas de audios de junio de 2017. En total, se dispone de 342 horas de grabaciones junto a sus anotaciones para realizar el trabajo, pero cabe destacar que luego del preprocesamiento de los datos esta cantidad se reduce.

Entre los eventos presentes en las anotaciones se tienen distintas vocalizaciones de ballenas azules, sonidos de barcos, sonidos provenientes de movimientos sísmicos, vocalizaciones de ballenas fin, vocalizaciones de ballenas minke, vocalizaciones de ballenas sei, entre otros.

Capítulo 4

Metodología

La metodología seguida para obtener modelos entrenados se puede resumir en seis etapas principales, que se presentan en el diagrama de flujo de la figura 4.1. Las seis etapas son preprocesamiento, división de la base de datos, extracción de $features$, creación de modelo HMM, determinación de modelo GMM y determinación de modelo DNN, en ese orden. A continuación, se presenta un resumen de estas etapas:

- **Preprocesamiento:** Dividir y limpiar grabaciones disponibles. Procesar anotaciones para ser utilizadas en los modelos.
- **División de base de datos:** Asignar grabaciones para el conjunto de entrenamiento y para el conjunto de *test*.
- **Extracción de** $features$**:** Dividir grabaciones en $frames$ y aplicar banco de filtros triangulares sobre la FFT de cada $frame$ para obtener vectores de características.
- **Determinación de modelo HMM:** Definir los símbolos observables, los estados del modelo y las posibles transiciones entre estados.
- **Determinación de modelo GMM:** Determinar gaussianas óptimas para calcular las probabilidades de observación en el modelo HMM.
- **Determinación de modelo DNN:** Ajustar red neuronal profunda para calcular las probabilidades de observación en el modelo HMM.

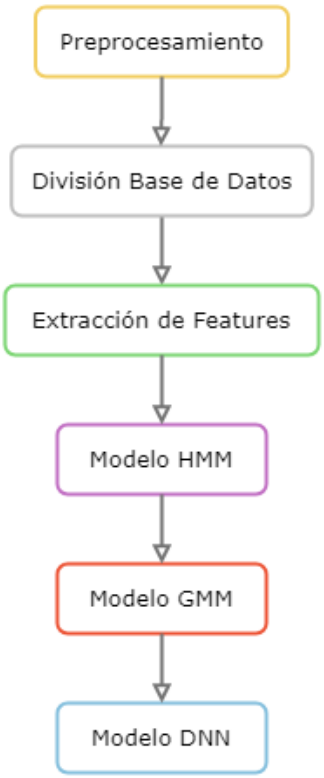

Figura 4.1: Diagrama de la metodología general seguida

4.1. Preprocesamiento

El preprocesamiento de los datos se divide en dos partes principales, el procesamiento de las grabaciones (ver rama izquierda de la figura 4.2) y el procesamiento de las anotaciones (ver rama derecha de la figura 4.2).

Figura 4.2: Diagrama de la metodología seguida para el preprocesamiento

4.1.1. Procesamiento de las grabaciones

En este trabajo se dispone de grabaciones que pueden tener distinto largo y la duración de estas suele ser de horas. El *software* Kaldi permite trabajar con señales de cualquier largo, pero suele funcionar mejor con señales de tamaño reducido que no superen los 1000 *frames*. Por esta razón, se procesan las grabaciones, generando señales de aproximadamente 600 *frames*. La lógica seguida para cortar las grabaciones se muestra en la figura 4.3.

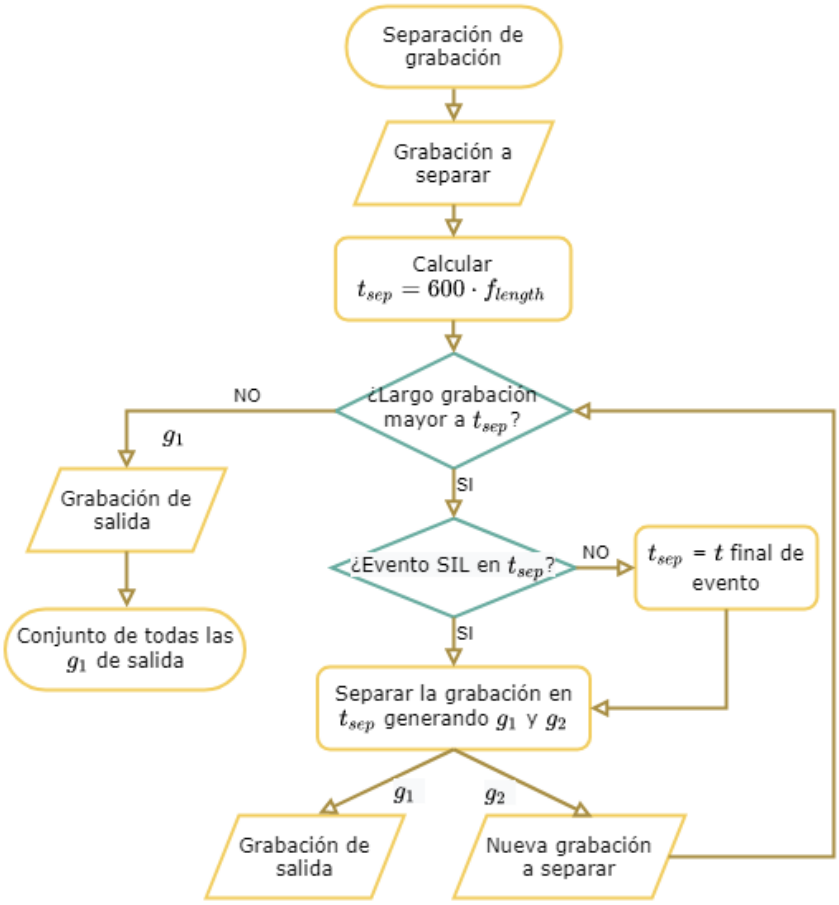

Figura 4.3: Diagrama lógico seguido para separar una grabación

En la figura 4.3 se muestra el procedimiento para separar una grabación en particular. La idea es ir separando la grabación original en subgrabaciones de 600 *frames* usando el tiempo $t_{sep} = 600 \cdot f_{length}$, que corresponde a un largo de 600 *frames* en segundos (aproximadamente 10 minutos de audio). Antes de separar en t_{sep} se verifica si se puede dividir lo que queda de grabación (que tenga duración mayor a t_{sep}). En caso de que no se pueda, se deja el resto de grabación actual como la última subgrabación y se termina el proceso. Si se puede dividir, se verifica que no haya un evento distinto de silencio en $t = t_{sep}$. En caso de que si haya un evento de interés, se aplaza t_{sep} hasta el final del evento, para evitar dividirlo. Una vez determinado t_{sep} se separa lo que queda de grabación en t_{sep}, el primer corte se considera como una nueva subgrabación y el segundo corte se toma como nueva grabación a dividir. Con este procedimiento algunas subgrabaciones generadas son de menor tamaño, ya que quedan al final de la grabación original y algunas grabaciones son de mayor tamaño, para evitar cortar en la mitad un evento en el final de la subgrabación. Una vez terminado el proceso,

se tiene un conjunto de subgrabaciones que divide la grabación original. Cabe destacar que este proceso se repite por cada grabación que se dispone.

Algunas grabaciones presentan ruido de saturación del hidrófono al inicio de la grabación y se elimina estas grabaciones, puesto que asignar la etiqueta SIL o UND a la saturación afecta los resultados.

Para entrenar cualquier modelo para la clasificación de distintas clases, es necesario tener suficientes muestras de cada clase, si no las clases con pocas muestras no serán reconocidas correctamente por el modelo. Por esta razón, se removió las clases con pocos ejemplos, en específico, se eliminó las etiquetas con menos de 10 $frames$ en total y/o que aparecen en menos de 3 grabaciones distintas. Se opta por eliminar las grabaciones donde aparecen estas etiquetas, puesto que son pocas y así se evita el problema de que etiqueta colocar en los $frames$ en que aparece. En la figura 4.4 se muestra la lógica seguida para eliminar las grabaciones con etiquetas deficientes. Se itera por cada etiqueta y al final del proceso se dispone de las grabaciones ya filtradas a utilizar en el trabajo.

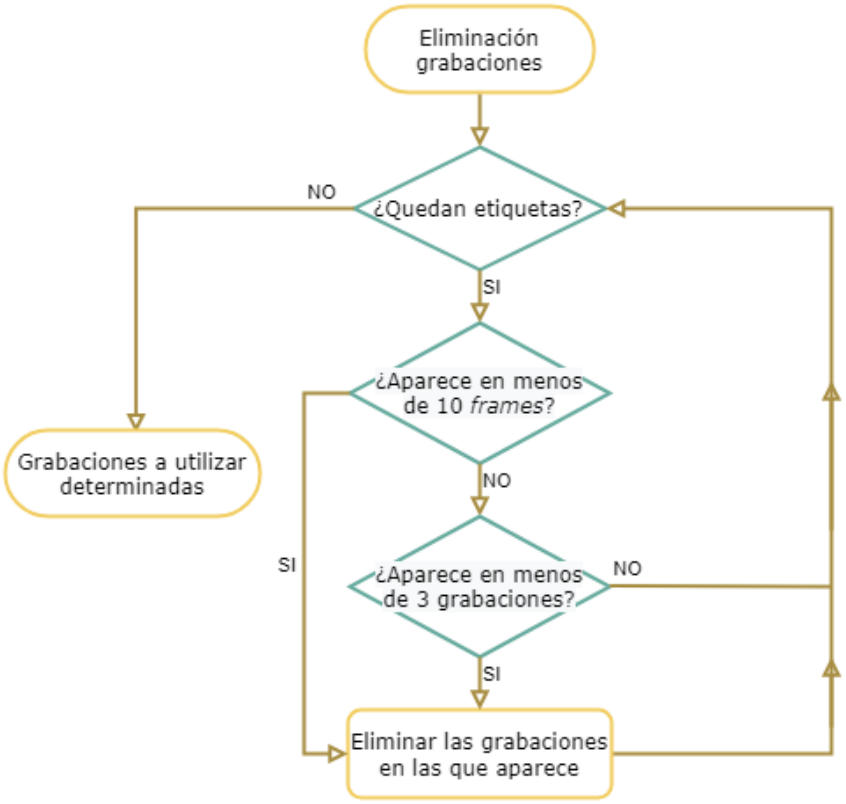

Figura 4.4: Diagrama lógico seguido para eliminar grabaciones de etiquetas con pocas muestras

Cabe destacar que el proceso de separación de las grabaciones genera algunas subgrabaciones compuestas solo de silencios. Estas grabaciones son eliminadas del *dataset* debido a la falta de información relevante para entrenar los modelos. Este paso, al igual que el proceso de eliminación de las grabaciones con etiquetas deficientes, reduce el tamaño de la base de datos disponible.

4.1.2. Procesamiento de las anotaciones

Otra tarea necesaria antes de entrenar el modelo corresponde a traducir las anotaciones a matrices adecuadas para usar en Kaldi. Las anotaciones contienen el tiempo de inicio, el tiempo de término y la etiqueta de los eventos dentro de la grabación asociada. Para usar Kaldi se requiere traducir estos tiempos a $frames$. Por esto, a partir de las anotaciones se crea la matriz ALI, que contiene un vector por cada grabación, correspondiente a la división de la grabación en $frames$. Se considera un solapamiento de $frames$ de 50 % por lo que, por ejemplo, si la grabación dura $100s$ y el tamaño del $frame$ es $10s$, entonces el vector en ALI tiene 19 elementos ($frames$). Cada elemento/$frame$ contiene la etiqueta de la anotación que cae dentro del $frame$. Cabe destacar que existen eventos solapados que durante cierto tiempo ocurren simultáneamente y se requiere crear una nueva etiqueta para este caso. Esto debido a que cada $frame$ tiene asociada una sola etiqueta y en el caso del evento solapado se tienen dos etiquetas, por lo que debe crear una tercera que denote el caso en que estos dos eventos ocurren al mismo tiempo.

Otro archivo necesario corresponde a la secuencia de etiquetas (transcripción) de cada grabación, que se obtiene fácilmente a partir de ALI. Aquí, se condensan las etiquetas iguales consecutivas en una etiqueta única. Por ejemplo, si el vector de una grabación en ALI es $A-A-A-C-A-B-B-D$, entonces la secuencia de estados es $A-C-A-B-D$. Este archivo es esencial para obtener alineamientos, como se verá más adelante.

Existen varios archivos extras que se deben crear para hacer el modelo funcionar y que no son mencionados por no ser esenciales para explicar el funcionamiento del trabajo.

4.2. División de la base de datos

Una vez realizado el preprocesamiento, se dispone de las grabaciones a utilizar en el proyecto, así como los archivos auxiliares necesarios para trabajar en Kaldi. Aquí, resulta necesario dividir la base de datos en conjuntos de entrenamiento y de *test*. Con los datos de entrenamiento se ajustan los modelos de manera que sea capaz de replicar correctamente las etiquetas de este conjunto y con los datos de *test* se prueba que tan bien clasifica el modelo sobre un conjunto de datos nuevos, para probar la capacidad de generalización de este. Los resultados relevantes son los porcentajes de clasificaciones correctas de *test*, puesto que indican la capacidad de clasificación del modelo sobre datos nuevos, tal como ocurre en una aplicación práctica.

En este trabajo se dividen los datos en 75 % de datos de entrenamiento y 25 % de datos para el conjunto de *test*. Realizar la división implica principalmente seleccionar que grabaciones asignar a cada conjunto y generar ciertos archivos auxiliares para Kaldi, de manera que el programa sepa que grabaciones son para entrenar y que grabaciones son para el *test*.

Resulta importante realizar una división balanceada entre los dos conjuntos, de manera que idealmente los porcentajes relativos de cada etiqueta se mantengan igual en ambos conjuntos. Realizar una división al azar puede resultar, por ejemplo, en que el conjunto de *test* contenga la mayoría de las etiquetas de cierta clase, lo que resultaría en un modelo mal

entrenado para esa etiqueta y resultados bajos de *accuracy* de *test* en consecuencia. Por esta razón, se toma la precaución de dividir de manera balanceada los datos, de forma que se tengan suficientes ejemplos de cada clase para entrenar el modelo y luego también para probarlo.

4.3. Extracción de $features$

Para utilizar las grabaciones para el entrenamiento de los modelos, resulta necesario extraer vectores de características ($features$) para representarlas. El proceso para obtener los vectores de $features$ se muestra en la figura 4.5. El primer paso es dividir cada grabación en $frames$. El largo de los $frames$ ($frame\ length$) es determinado a partir de la frecuencia de muestreo con la que se generó la señal (f_s) y a partir del tamaño de la ventana usada para obtener las FFT ($FFT\ size$). El largo de los $frames$ sigue la siguiente ecuación:

$$frame\ lenght = \frac{FFT\ size}{f_s}$$

El tamaño de ventana usado para obtener la FFT es de 512 muestras y la f_s de las grabaciones es $250Hz$, por lo que el largo de cada $frame$ es $2.048s$. Cabe destacar que se usó un solapamiento del 50 %, por lo que cada $frame$ parte en la mitad del $frame$ anterior.

El siguiente paso es obtener una representación en el dominio de la frecuencia de cada $frame$. Para esto primero se aplica una ventana de Hamming sobre los $frames$ de la señal, para así reducir los lóbulos laterales que aparecen al calcular la FFT debido al fenómeno de manchado espectral (*spectral leakage*). Luego se calcula la FFT de cada $frame$ y se rescata el valor absoluto de esta (no se utiliza la fase).

Posteriormente se aplica el banco de filtros de frecuencias de la figura 4.6 sobre las FFT, con lo que se obtiene un vector de N_{filter} (número de filtros del banco) componentes por cada $frame$. Se aplica la función logaritmo sobre cada componente.

En base al procedimiento explicado, se dispone de $features$ estáticos en el sentido de que cada $frame$ tiene una descripción independiente de los $frames$ de tiempos anteriores y posteriores. Para mejorar la descripción se agregan los coeficientes Δ y $\Delta\Delta$, que se pueden entender como la derivada y la segunda derivada evaluada (discretas) en cada $frame$/tiempo respectivamente.

Los coeficientes Δ se calculan a partir de la ecuación:

$$\Delta(t) = \frac{2 \cdot s(t+2) + s(t+1) - s(t-1) - 2 \cdot s(t-2)}{10}$$

Donde $s(t)$ corresponde al vector de $features$ estático del $frame\ t-$ésimo.

Las segundas derivadas discretas se calculan a partir de las primeras derivadas siguiendo la ecuación:

$$\Delta\Delta(t) = \frac{2 \cdot \Delta(t+2) + \Delta(t+1) - \Delta(t-1) - 2 \cdot \Delta(t-2)}{10}$$

Finalmente se normalizan los vectores de características con MVN (Normalización de Media y Varianza). De esta manera, la media de cada vector se traslada a 0 y la varianza de cada vector queda normalizada en 1.

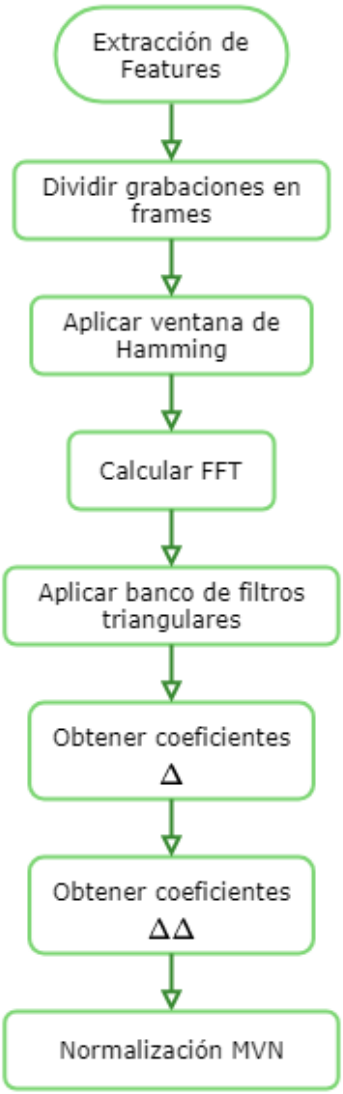

Figura 4.5: Diagrama de la metodología seguida para la extracción de $features$

Considerando un banco de filtros de N_{filter} filtros y la división de una grabación en N_{frames} $frames$, entonces la grabación queda descrita por una matriz de características de $3 \cdot N_{filter}$ x N_{frames}.

Al repetir este proceso sobre cada una de las N_{grab} grabaciones, se tiene que el *dataset* completo queda descrito por una matriz de dimensiones N_{grab} x $3 \cdot N_{filter}$ x N_{frames}, donde cada $frame$ de cada grabación tiene asociado un vector de N_{filter} componentes. Esta matriz de características se denomina matriz $Feats$ en este trabajo.

4.3.1. Banco de filtros triangulares

Para esta aplicación se utiliza un banco de filtros triangulares, donde la mayor densidad de filtros se encuentra en las frecuencias bajas. Esto se debe a las características de las señales utilizadas, donde los objetos son de baja frecuencia, generalmente menores a 100 Hz.

Cada filtro F_i está centrado en la frecuencia f_i^c y tiene un ancho de banda B_i, siguiendo la siguiente ecuación:

$$F_i = \begin{cases} -\frac{2}{B_i}|f - f_i^c| + 1 & si \ \ |f - f_i^c| \leq \frac{B_i}{2} \\ 0 & si \ \ |f - f_i^c| > \frac{B_i}{2} \end{cases}$$

Los filtros triangulares fueron creados de manera que la frecuencia central de un filtro calce con la frecuencia de corte menor del filtro siguiente. A la vez, el ancho de banda de los filtros se ideó para que bajo una frecuencia umbral f_{th} el ancho se mantenga en un valor constante B y, a partir de f_{th}, comience a aumentar linealmente conforme a un parámetro α. La forma del banco de filtros resultante se presenta en la figura 4.6

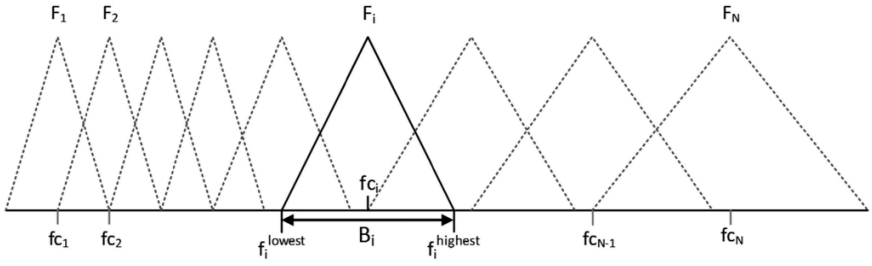

Figura 4.6: Banco de filtros de frecuencia de forma triangular

Los anchos de banda del banco utilizado pueden ser descritos matemáticamente por la siguiente función:

$$B_i = \begin{cases} B & si \ \ f_{i-1} - f_{th} \leq 0 \\ B + (f_{i-1} - f_{th}) \cdot tan(\alpha) & si \ \ f_{i-1} - f_{th} > 0 \end{cases}$$

Con ancho de banda inicial $B_1 = B$. Las frecuencias centrales de los filtros están dadas por la ecuación:

$$fc_i = fc_{i-1} + \frac{B_i}{2}$$

Con frecuencia central inicial $fc_1 = \frac{B}{2}$.

Como se puede ver, se requieren tres parámetros necesarios para definir completamente el banco de filtros:

- **Frecuencia umbral f_{th}:** Determina en que frecuencia el ancho de banda de los filtros comienza a aumentar. Un valor alto de f_{th} no cambia la definición entre el rango bajo y alto de frecuencias, en cambio, un valor bajo de f_{th} aumenta la definición en el rango bajo de frecuencias.

- **Parámetro α:** Determina la magnitud del aumento lineal de los anchos de banda después de la frecuencia umbral. Un valor alto de α marca una gran diferencia en definición antes y después de la frecuencia f_{th}. Con un valor bajo de α la diferencia en definición antes y después de la frecuencia f_{th} no es tan marcada.
- **Parámetro N:** Determina el número de $filtros$ totales en el banco de filtros. Notar que una vez definidos f_{th}, α y N, el parámetro B queda fijo, por lo que no se considera como un parámetro extra.

Los valores óptimos de los parámetros f_{th}, α y N se deben determinar empíricamente, por lo que se realizaran experimentos con distintas combinaciones de valores para determinarlos. Es importante mencionar que el valor de α se fija en $\alpha = 30°$ para minimizar el espacio de búsqueda, por lo que sólo se modifica f_{th} y N.

4.3.2. HMM

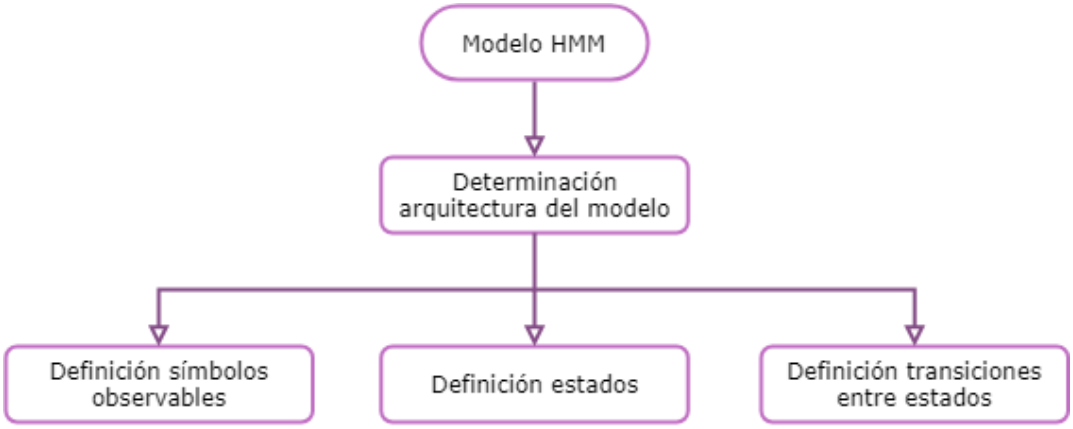

Figura 4.7: Diagrama de la metodología seguida para la definición del modelo HMM

El núcleo del modelo utilizado para la detección y clasificación es *Hidden Markov Model*, por lo que se debe construir la tupla $M = \{S, \sum, A, B, \Pi\}$ para tener un modelo correctamente definido. Recordemos que en el modelo M, los parámetros {S, $\sum$, A, B, Π} son:

- S: El conjunto finito de estados del modelo, conocido como espacio de estados.
- $\sum$: El conjunto finito de los posibles símbolos a observar.
- A: La matriz de probabilidades de transición entre estados de S.
- B: La matriz de probabilidad de emisión de símbolos de $\sum$, también conocidas como probabilidades de observación.
- Π: El vector de probabilidades iniciales.

4.3.3. Símbolos observables

Como se mencionó en la sección de preprocesamiento, se dispone de una base de posibles etiquetas a partir de las anotaciones entregadas. Esta base contiene directamente las etiquetas presentes en las anotaciones y a esta base se agregan además las etiquetas de eventos solapados que aparecen. Por otra parte, se eliminan de esta base las etiquetas con muy pocas muestras. Así, se define el conjunto Σ de los símbolos observables como el conjunto resultante de etiquetas, luego de agregar las etiquetas de eventos solapados y de eliminar las etiquetas con pocos ejemplos.

Tomando en consideración lo anterior se obtuvo la lista de etiquetas presentada en la tabla 4.1. Como se puede ver, hay 17 etiquetas que denotan eventos individuales sin solapamiento y también aparecen 7 etiquetas asociadas a eventos solapados. La nomenclatura de etiquetas es bastante explicativa, si en un $frame$ cae dos anotaciones distintas, una con etiqueta A y otra con etiqueta B, entonces aparece una nueva etiqueta AB que se asocia a este $frame$. En la tabla se muestra el número de la etiqueta, que es la verdadera etiqueta usada para denotar al evento dentro de los modelos.

Tabla 4.1: Etiquetas consideradas para la clasificación

N°	Etiqueta	Evento
1	SIL	Silencio
2	UND	Indefinido
3	ERQ	Sismo
4	S13	Southeast Pacific 1 Unidad C
5	S21	Southeast Pacific 2 Unidad A
6	S22	Southeast Pacific 2 Unidad B
7	S23	Southeast Pacific 2 Unidad C
8	SEP	Southeast Pacific indefinido
9	FWS	Fin Whale Song
10	FWD	Fin Whale Downsweep tipo 1
11	FWD2	Fin Whale Downsweep tipo 2
12	FWD3	Fin Whale Downsweep tipo 3
13	SWD	Sei Whale Upsweep
14	SWU	Sei Whale Downsweep
15	MI	Minke Whale
16	SHIP	Barco
17	AA	Antarctic Blue Whale
18	UNDS22	Indefinido + Southeast Pacific 2 B
19	UNDAA	Indefinido + Antarctic Blue Whale
20	ERQFWS	Sismo + Fin Whale Song
21	S22FWS	Southeast Pacific 2 B + Fin Whale Song
22	S23FWS	Southeast Pacific 2 C + Fin Whale Song
23	FWSFWD	Fin Whale Song + Fin Whale Dowsweep 1
24	AAFWS	Antartic Blue Whale + Fin Whale Song

4.3.4. Espacio de estados

El modelo HMM es una máquina de estados finitos y como tal resulta necesario definir los estados en los que la máquina puede estar. La arquitectura del HMM utilizado se presenta en la figura 4.8 y corresponde a un HMM con tres estados por cada posible etiqueta a clasificar. Así, la ocurrencia de cualquier evento es modelada por su correspondiente *set* de 3 estados donde el primer, segundo y tercer estado corresponden al inicio, el medio y el final del evento respectivamente. Notar que esta definición de los estados depende directamente de la base

de etiquetas posibles de la tabla 4.1.

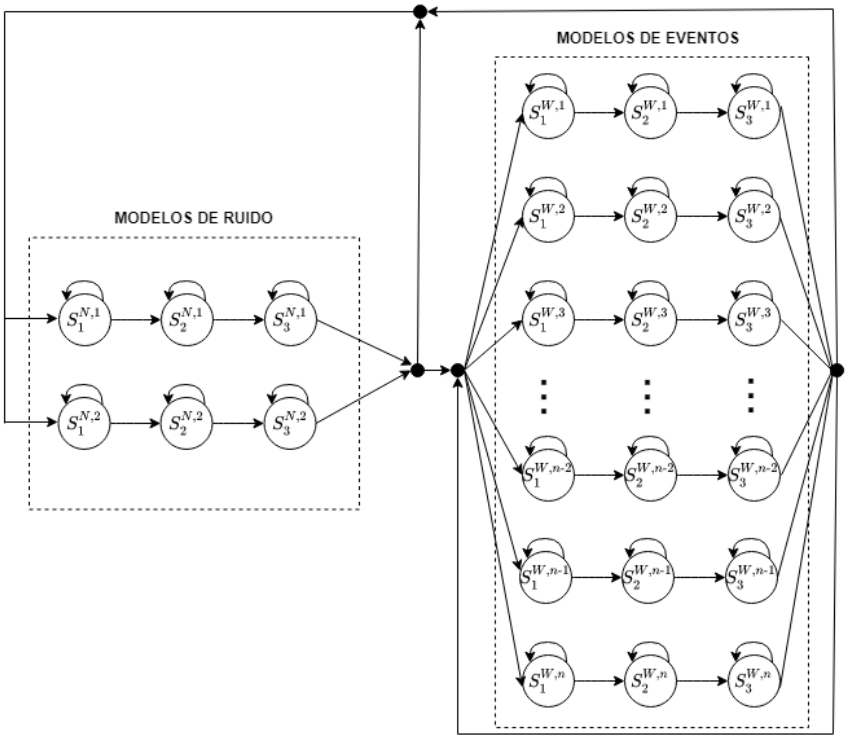

Figura 4.8: Arquitectura de modelo HMM sin salto desde el primer estado

En la figura 4.8 se puede ver la arquitectura implementada, donde se diferencia entre los modelos de silencio, asociados a las etiquetas SIL y UND, con las etiquetas de eventos restantes. En los modelos de eventos, n corresponde al número total de estos eventos (distintos de SIL y UND). Se puede notar que en los modelos de eventos el primer estado solo puede moverse hacia si mismo o al segundo estado, el segundo estado puede moverse hacia si mismo o hacia el tercer estado y el tercer estado puede moverse hacia si mismo o salir al primer estado de cualquier modelo. Por esta razón, una vez se está en el primer estado, se debe pasar por tres estados asociados al evento correspondiente, lo que puede ser un problema si se tienen eventos de menos de 3 $frames$. Esto motiva a incluir un tercer posible movimiento para el primer estado, que permita salir del *set* de 3 estados del evento. Agregando este salto desde el primer estado se tiene un modelo HMM como el de la figura 4.10. Como se puede ver, no todos los eventos son modelados con este salto, la lógica seguida para determinar el modelo se muestra en la figura 4.9. Siguiendo este procedimiento, se itera por cada una de las etiquetas de la tabla 4.1, se crea un *set* de 3 estados asociados a cada etiqueta y se agrega un salto desde el primer estado solo a las etiquetas con eventos de largo menor a tres $frames$. Al terminar la iteración por cada una de las etiquetas, se tiene definido un modelo como el de la figura 4.10, donde n_1 es el número de estos eventos distintos de SIL y UND a los que se agrega el salto y n_2 es el número de los eventos distintos de SIL y UND a los que no se agrega el salto.

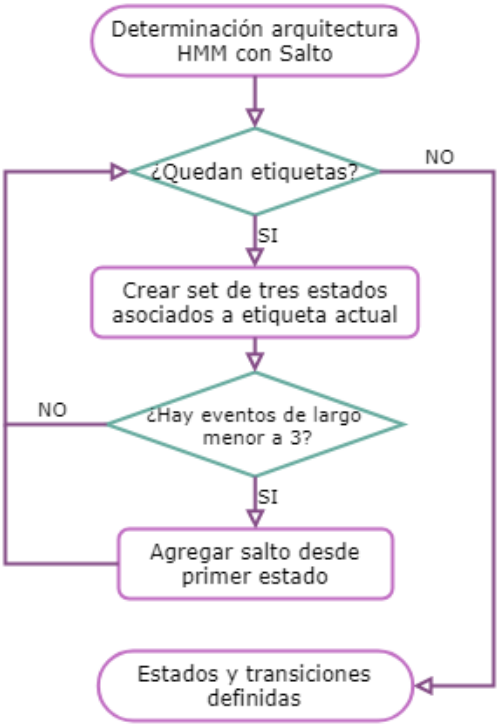

Figura 4.9: Diagrama lógico seguido para la determinación del modelo HMM con salto desde el primer estado

Con este modelo HMM es posible modelar cualquier secuencia de eventos que puedan aparecer en una grabación, ya que el final de un evento puede saltar al inicio de cualquier otro evento. Cabe destacar que se utilizan los modelos de las figuras 4.8 y 4.10 para obtener resultados y realizar comparaciones de los porcentajes de clasificaciones correctas con o sin el salto desde el primer estado.

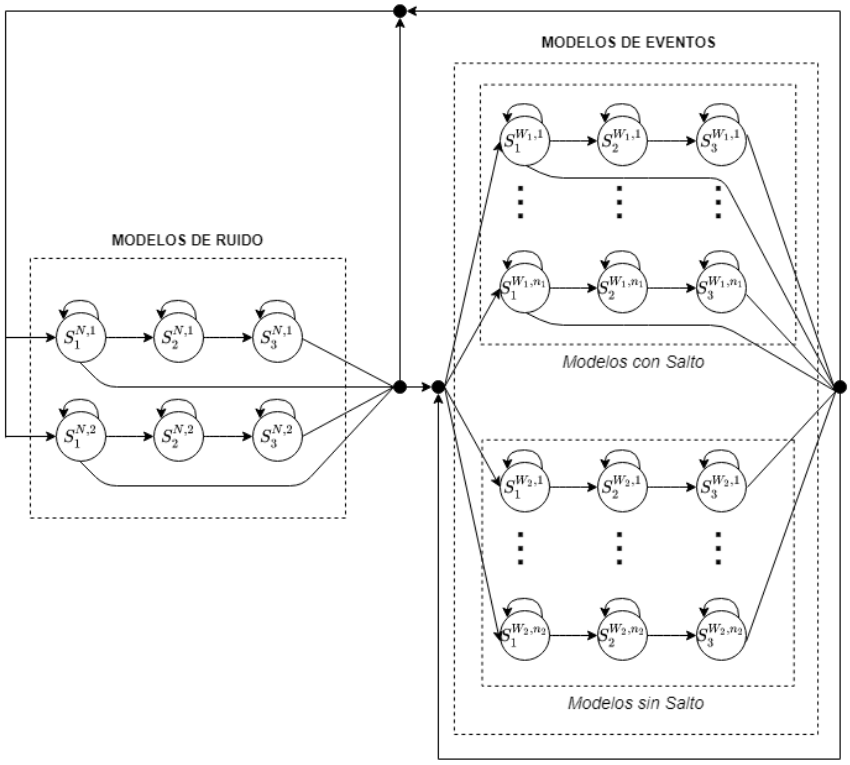

Figura 4.10: Arquitectura de modelo HMM con salto desde el primer estado

4.3.5. Probabilidades de transición

En los modelos de las figuras 4.8 y 4.10 cada una de las flechas corresponde a una transición de estado. La probabilidad de pasar del estado S_i al estado S_j corresponde a $A[i, j]$, donde las probabilidades en la matriz A son inicializadas manualmente. La suma de las probabilidades de transición desde un mismo estado es 100 %, para que el modelo este definido correctamente.

Cabe destacar que estas probabilidades de transición del modelo son ajustadas durante la fase de entrenamiento por Kaldi, por lo que son valores determinados internamente por el $software$ para maximizar los resultados entregados.

4.3.6. Probabilidades iniciales

El modelo HMM requiere probabilidades iniciales definidas, que en este caso corresponden a la probabilidad de que cada estado sea asignado al primer $frame$ de un evento. Por lo mismo, el largo de este vector de probabilidades iniciales Π está relacionado con el número de estados iniciales de la HMM (primeros estados en cada *set* de tres estados). Este vector Π puede ser visto como la distribución de probabilidad del primer estado, por lo que la suma de sus elementos corresponde a 100 %. Cabe destacar que los valores de estos parámetros son determinados internamente por Kaldi y son ajustados en el entrenamiento para entregar un mejor resultado.

4.3.7. Probabilidades de observación

Otro parámetro necesario para definir una HMM son las probabilidades de observación, que en este caso representan la probabilidad de observar cierta etiqueta estando en cierto estado de la HMM (tomando un $frame$). Para calcular estas probabilidades se utilizó dos métodos distintos, primero se implementa un modelo de mezcla de gaussianas (GMM) y luego se prueba usando una red neuronal profunda en su lugar. Estas dos formas de calcular las probabilidades de observación son de especial importancia y se explican en detalle a continuación. Cabe destacar que se obtienen resultados usando ambos métodos y se comparan los resultados para determinar qué modelo funciona mejor para este problema en particular.

4.4. Gaussian Mixture Model

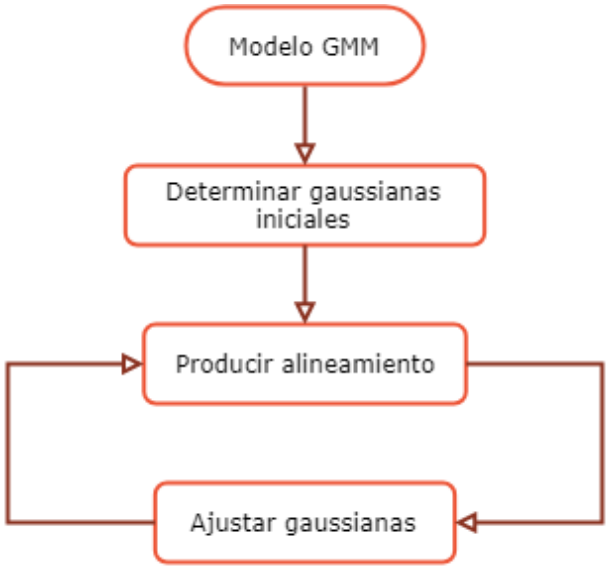

Figura 4.11: Diagrama de la metodología seguida para el entrenamiento del modelo GMM

Para utilizar el modelo oculto de Markov, resulta necesario calcular las probabilidades de observación a partir de las grabaciones. Para obtener estas probabilidades de observación a partir de un $frame$ se utiliza un modelo de mezcla de gaussianas GMM.

En este problema en particular, cada estado tiene un conjunto de gaussianas asociadas ($3 \cdot N_{filter}$ gaussianas). La idea es que cada $frame$ está representado por un vector de características de $3 \cdot N_{filter}$ componentes y cada posible estado tiene un conjunto de $3 \cdot N_{filter}$ gaussianas característico en consecuencia, por lo que se puede obtener una probabilidad de observación para cada estado evaluando el vector de $features$ del $frame$ en el conjunto de gaussianas de cada estado.

Así las probabilidades de observación son calculadas usando el modelo GMM definido como:

$$g(X, \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(X - \mu_i)' |\Sigma_i|^{-1} (X - \mu_i)\right\} \tag{4.1}$$

En la ecuación, i denota el estado en el que se está calculando la probabilidad de observación, o sea $g(X, \mu_i, \Sigma_i) = P(X|S = S_i)$, donde i va desde 1 a $N_{estados}$. El vector μ_i es el vector de medias de las gaussianas para el estado $i - ésimo$, Σ_i es la matriz de covarianza de las gaussianas del estado $i - ésimo$ y X es el vector de $features$ del $frame$ considerado. La dimensionalidad de los datos corresponde a D, con $D = 3 \cdot N_{filter}$, los vectores μ_i y X tienen dimensión D y la matriz Σ_i es de dimensión D x D. Cabe destacar que se consideran matrices de covarianza diagonales, de la forma:

$$\Sigma_i = \begin{pmatrix} \sigma_1^i & 0 & \cdots & 0 \\ 0 & \sigma_2^i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{3N_{filter}}^i \end{pmatrix}$$

Donde σ_j^i es la varianza de la gaussiana asociada al componente $j - ésimo$ de los $features$, considerando el estado $i - ésimo$ del modelo HMM. Así, se tienen $3 \cdot N_{filter}$ parámetros para definir la matriz Σ_i de cada estado.

En resumen, las probabilidades de observación de cada estado quedan definidas por $2 \cdot D = 6 \cdot N_{filter}$ parámetros. En total, se tienen $3 \cdot N_{etiq}$ estados, por lo que el modelo GMM completo requiere la determinación de $18 \cdot N_{etiq} \cdot N_{filter}$ parámetros.

A continuación se explica el proceso para determinar los parámetros de las gaussianas (medias y varianzas), de manera que la salida del modelo se ajuste a los datos.

4.4.1. Determinación de parámetros de las gaussianas iniciales

Para obtener los parámetros de las gaussianas se requieren dos cosas, un alineamiento y la matriz $Feats$. El alineamiento es la secuencia de etiquetas que aparecen en la grabación, $frame$ por $frame$. Para explicar esto más fácilmente se presenta el espectrograma de una grabación de ejemplo (ver figura 4.12). La grabación está compuesta de silencio (SIL), seguido de un evento sísmico (ERQ) y luego más silencio. Aquí claramente la secuencia (transcripción) es *SIL-ERQ-SIL*, pero para tener un alineamiento falta determinar las etiquetas $frames$ por $frame$. En el espectrograma de ejemplo de la figura 4.12 se toman $frames$ de 10 segundos sin solapamiento (delimitados por líneas rojas) y en este caso se tiene el alineamiento *SIL-SIL-SIL-ERQ-ERQ-ERQ-ERQ-ERQ-ERQ-SIL-SIL-SIL*. Esta información es conocida a priori para el entrenamiento, ya que está presente en la matriz ALI descrita en la sección de Preprocesamiento, por lo que a partir de los datos de entrenamiento uno posee un alineamiento inicial a utilizar.

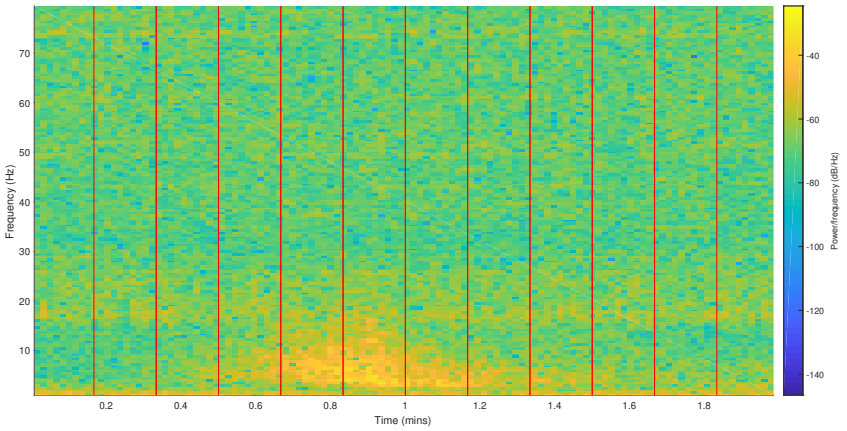

Figura 4.12: Espectrograma de señal de ejemplo con un evento sísmico, usando $frames$ de $10s$

Para calcular la media y varianza de cada una de las gaussianas se utiliza el alineamiento para saber qué $frames$ corresponden a una etiqueta en particular. Luego se obtiene la media y la varianza de los $features$ de estos $frames$ a partir de la matriz $Feats$. Por ejemplo, para el caso de la figura 4.12, la etiqueta ERQ tiene asociado desde el $frame$ 4 hasta el 9, por lo que se deben extraer los vectores de $features$ asociados a estos seis $frames$ de la matriz $Feats$, obteniéndose una matriz de 6 x $3 \cdot N_{filter}$. Luego, se divide el evento en tres partes iguales (parte inicial, media y final del evento), cada una asociada a uno de los tres estados de su *set* en el modelo HMM. Así, se tendrá una matriz de 2 x $3 \cdot N_{filter}$ para cada uno de los tres estados. Posteriormente, se calcula la media y la varianza a partir de la primera dimensión, obteniéndose $3 \cdot N_{filter}$ medias y $3 \cdot N_{filter}$ varianzas para cada estado del *set*, lo que define sus $3 \cdot N_{filter}$ gaussianas asociadas. Este procedimiento se itera para todas las etiquetas y así se definen las gaussianas iniciales de los estados (tres estados del *set* correspondientes en el modelo HMM) de todas las etiquetas.

4.4.2. Ajuste de las gaussianas

El proceso de entrenamiento y ajuste de las gaussianas se muestra en la figura 4.11. Primero se calcula un nuevo alineamiento a partir del *set* actual de gaussianas y la secuencia de etiquetas. Para obtener el nuevo alineamiento se utilizan las gaussianas actuales para calcular las probabilidades de observación del modelo HMM según la ecuación 4.1 y se aplica el algoritmo de Viterbi, considerando que se debe cumplir con la secuencia de etiquetas de entrenamiento (transcripción de las grabaciones). Así, se obtiene un nuevo alineamiento, que es la secuencia más probable según el algoritmo de Viterbi (usando las probabilidades de observación dadas por las gaussianas actuales).

Una vez obtenido el nuevo alineamiento, se utiliza el mismo método descrito en la subsección anterior para obtener nuevas medias y varianzas a partir de este nuevo alineamiento y de la matriz $Feats$. Las gaussianas ajustadas corresponden a gaussianas con estas nuevas medias y varianzas como parámetros.

Este proceso de refinamiento de las gaussianas del modelo GMM (refinamiento de las probabilidades de observación del modelo HMM), se repite 40 veces como parte del entrenamiento. Al final del entrenamiento se dispone de un modelo GMM que permite calcular las probabilidades de observación de manera que las predicciones del modelo HMM se ajusten a los datos de entrenamiento.

4.5. Deep Neural Network

En la figura 4.13 se muestra el procedimiento seguido para entrenar una red neuronal profunda para calcular las probabilidades de observación del modelo HMM. Primero se entrena un *autoencoder* para representar los datos disponibles de una forma óptima, luego se define los parámetros de la red y se comienza a entrenar. Para el entrenamiento inicialmente se utiliza el mejor alineamiento obtenido con el modelo GMM y a partir de este alineamiento se ajustan los pesos de una primera DNN. Luego, usando la red recién entrenada se produce un nuevo alineamiento, con el que se entrena una siguiente red neuronal profunda. Este proceso se repite cuatro veces, ajustando los parámetros de 4 redes DNN. Cabe destacar que estas redes no se agregan, sino que reemplazan a la red neuronal anterior.

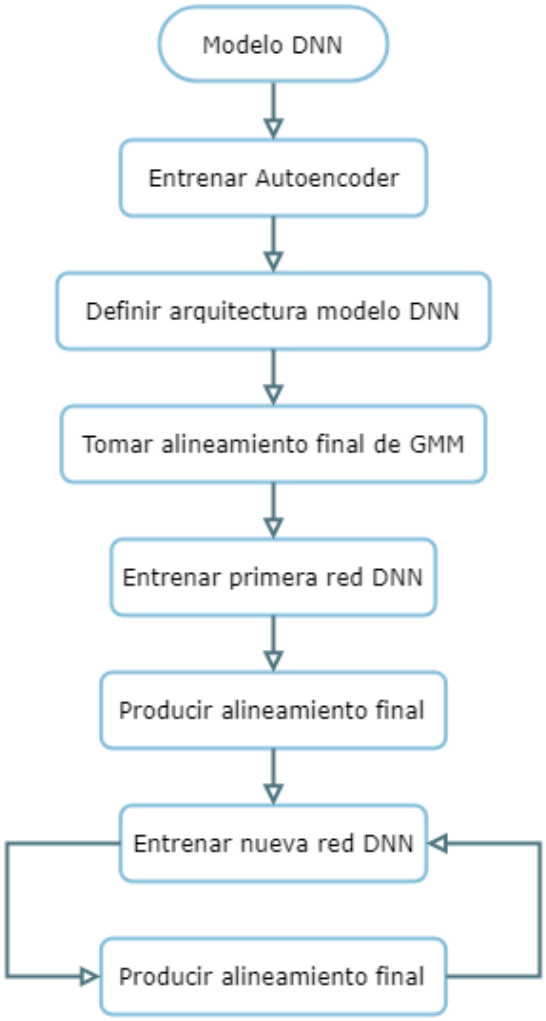

Figura 4.13: Diagrama de la metodología seguida para el entrenamiento de las redes neuronales profundas

4.5.1. Pre-entrenamiento de la red DNN (*Autoencoder*)

Uno de los problemas que surgen al utilizar redes neuronales de varias capas es la posible perdida de información al ir avanzando hasta las capas finales de la red. Una inicialización al azar de los pesos puede resultar en una red inicial donde prácticamente no llega información relevante a la capa final. En este caso el proceso de *backpropagation* en el algoritmo de gradiente descendiente no puede cambiar adecuadamente los pesos de las primeras capas de la red y el proceso de entrenamiento resulta defectuoso. Para evitar este problema y obtener una DNN inicial que utilice al máximo la información presente en los datos, se genera la configuración inicial de la red entrenando redes *autoencoder*.

Para obtener la estructura inicial de una red DNN con N_{layers} capas, se entrenan $N_{layers}-1$ *autoencoders* y se utiliza la capa de codificación de estos para definir los pesos iniciales de la red DNN. El proceso consiste en entrenar un primer *autoencoder* usando los datos de entrada, luego entrenar un segundo *autoencoder* usando la salida del primer *autoencoder*, posteriormente entrenar un tercer *autoencoder* usando la salida del segundo *autoencoder* (sa-

lida resultante de la concatenación del primer y segundo *autoencoder*) y así hasta entrenar $N_{layers} - 1$ $autoencoders$. Finalmente se toman los pesos de la capa de codificación de cada *autoencoder* y se inicializa la última capa de la red DNN al azar, con lo que se define la estructura inicial de la red deseada. Este proceso asegura que la información de los datos llegue al final de la red, puesto que la configuración inicial permite recuperar los datos de entrada en las capas finales de esta.

Los *autoencoder* utilizados en este trabajo se asemejan al presentado en la figura 2.28. El número de neuronas de codificación del primer *autoencoder* define el tamaño de los nuevos $features$ por cada $frame$ de la nueva representación. A la vez, se utiliza el mismo número de neuronas de codificación en todos los $autoencoders$, por lo que la capa de codificación del primer *autoencoder* define el número de neuronas en las capas de la red DNN.

La cantidad de neuronas de codificación a utilizar para obtener una buena representación de los datos está relacionada con el tamaño original de los vectores de $features$. Normalmente se usa un número de codificación menor al tamaño original de los $features$ para obtener representaciones más compactas y con menos ruido. A la vez, no se puede usar un número demasiado bajo para la codificación, puesto que se obtienen representaciones sin suficiente información para la clasificación. Por esta razón, en este trabajo se prueba usando un número de neuronas de codificación del mismo tamaño, cuatro tercios del tamaño y un medio del tamaño de los vectores de $features$ originales. Así, si se usa un banco de filtros que genera vectores de $features$ de N_f componentes, se prueba entrenando tres *autoencoder*, con N_f, $\frac{3}{4}N_f$ y $\frac{1}{2}N_f$ neuronas de codificación, para ver que representación entrega mejores resultados.

4.5.2. Estructura del modelo DNN

La red neuronal profunda a implementar queda definida por dos parámetros, la cantidad de capas de la red (N_{layers}) y la cantidad de neuronas en las capas ocultas (N_{hidden}). Para reducir el número de parámetros a optimizar, se usan capas ocultas con el mismo número de neuronas. A la vez, se usa como valor de N_{hidden} el mismo número de neuronas de codificación del *autoencoder*, por lo que la búsqueda de la mejor representación esta relacionada con la búsqueda del número óptimo de neuronas ocultas. En la figura 4.14 se muestra un ejemplo de red DNN con $N_{layers} = 5$ y $N_{hidden} = 4$, se puede ver como el número de neuronas de entrada concuerda con el número de neuronas ocultas, puesto que se usa N_{hidden} igual al número de neuronas de codificación del *autoencoder*. La red de ejemplo posee 3 neuronas de salida, en las redes DNN de este trabajo se tienen 72 neuronas para la capa de salida correspondientes a los 72 estados de la red HMM (tres veces el número de etiquetas). Cabe destacar que las cuatro redes DNN entrenadas comparten la misma estructura definida inicialmente. Los parámetros N_{layers} y N_{hidden} óptimos deben ser determinados experimentalmente, para esto, se realiza una búsqueda de malla en el espacio de los parámetros en busca del mayor porcentaje de clasificaciones correctas posible.

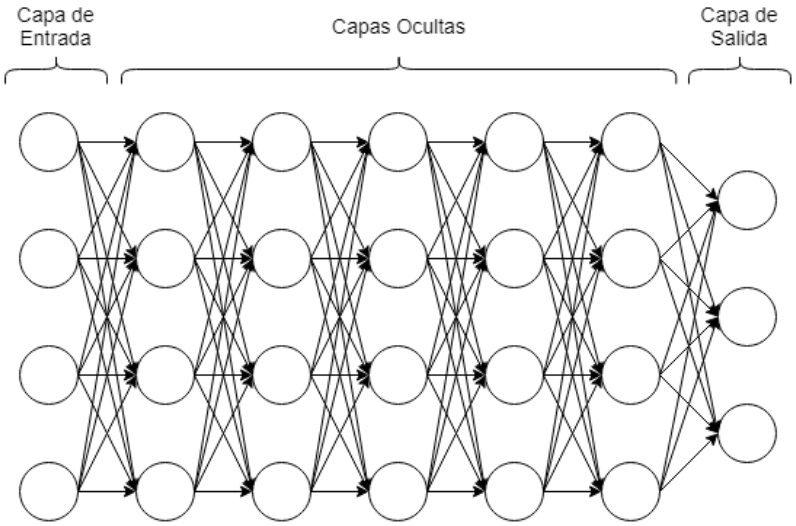

Figura 4.14: Red DNN de ejemplo con $N_{layers} = 5$ y $N_{hidden} = 4$

4.5.3. Primera red neuronal profunda

Para entrenar una red neuronal profunda se requieren tres cosas: la matriz de $features$ de las grabaciones (resultante de la representación usando el *autoencoder*), la transcripción de cada grabación (actúan como etiquetas correctas para el entrenamiento) y un alineamiento inicial. Para la primera red DNN se utiliza el mejor alineamiento obtenido con el modelo GMM como alineamiento inicial. Es por esta razón que en diagrama de la figura 4.1 la obtención del modelo DNN depende del modelo GMM, aun cuando cumplen con la misma función.

La definición y el entrenamiento de la DNN se realiza en el *software* Kaldi. El objetivo del entrenamiento en Kaldi es que las transcripciones de la salida del modelo entrenado concuerden con las transcripciones originales de las grabaciones. Así, el entrenamiento no está orientado a calzar las etiquetas $frame$ a $frame$, sino que está orientado a calzar las secuencias de eventos.

4.5.4. Segunda, tercera y cuarta DNN

Para refinar los resultados obtenidos usando una red DNN, se utiliza la misma red preentrenada, pero en este caso se entrena con el alineamiento final obtenido con la red a refinar. Así, la diferencia entre la red y la red siguiente radica sólo en el alineamiento inicial, donde la red siguiente inicia desde un alineamiento generalmente de mejor calidad que permite obtener mejores resultados.

El proceso de refinamiento de la red DNN entonces se compone de un *loop* en el que se entrena una red, se obtiene un alineamiento al final del entrenamiento y se utiliza este alineamiento como alineamiento inicial para la siguiente red DNN. En este trabajo se repite este ciclo 4 veces, de manera que al final del proceso de refinación se obtienen 4 redes DNN de la misma estructura, pero con pesos distintos y distintos resultados de *accuracy*. Así, para una determinada estructura de DNN con ciertos parámetros N_{layers} y N_{hidden}, se tienen 4 *accuracies* de *test* (uno por cada proceso de refinamiento) de la que se toma el valor más

alto como el resultado de *test* para esta determinada combinación de parámetros.

4.6. Método de cálculo de resultados

Para determinar qué modelo entrenado es el mejor, resulta necesario definir claramente que métricas se consideraran. Este trabajo se enfoca en obtener el mayor porcentaje de clasificaciones correctas posible considerando todas las etiquetas a la vez, o sea, se busca que el modelo acierte la mayor cantidad de veces a las etiquetas correctas en general. Para calcular el porcentaje de clasificaciones correctas, se compara la etiqueta entregada por el modelo en cada $frame$ con la etiqueta correcta clasificada por los expertos. La fórmula usada para el cálculo es:

$$accuracy = 100 \cdot \frac{\sum_{i=1}^{N_{fr}} [A_i = B_i]}{N_{fr}}$$

Donde N_{fr} corresponde al número total de $frames$ en el conjunto de *test*, A_i es la etiqueta del $frame\ i-$ésimo entregada por el modelo entrenado y B_i es la etiqueta del $frame\ i-$ésimo correcta, definida inicialmente por los expertos. Cabe destacar que en la fórmula N_{fr} considera todas las grabaciones del conjunto de *test*. La función corchete se define como:

$$[A_i = B_i] = \begin{cases} 1 & si\ \ A_i = B_i \\ 0 & si\ \ A_i \neq B_i \end{cases}$$

Es importante destacar que esta forma de calcular el resultado implica que se busca un modelo que maximice los aciertos totales, por lo que se intrínsecamente da mayor relevancia a las etiquetas con mayor número de ejemplos. Así, etiquetas con pocos ejemplos en los *dataset* influyen poco en el resultado final entregado por los modelos.

Para cuantificar los resultados obtenidos con las etiquetas menos numerosas se calcula el $recall$ de los experimentos. El $recall$, también denominado sensibilidad, en problemas de clasificación binaria, entrega el porcentaje de las etiquetas positivas correctamente clasificadas por el modelo entre todas las etiquetas realmente positivas. En términos de estadísticos se calcula con la fórmula:

$$recall = \frac{TP}{TP + FN}$$

Donde TP son los verdaderos positivos (etiquetas que son positivas y son clasificadas como positivas por el modelo) y FN son los falsos negativos (etiquetas que son positivas y son clasificadas como negativas por el modelo).

Claramente esta métrica no se puede utilizar directamente en problemas de clasificación con múltiples clases, como es el caso de este trabajo. Para utilizarla se define como clase negativa el evento silencio y como clase positiva el resto de los eventos que si contienen información relevante. Así el $recall$ de los experimentos en este caso se calculan a partir de la ecuación:

$$recall = 100 \cdot \frac{\sum_{j=2}^{N_{etiq}} \sum_{i=1}^{N_j} [A_i^j = j]}{\sum_{j=2}^{N_{etiq}} N_j}$$

Donde N_{etiq} es el número total de etiquetas, N_j es el número total de $frames$ clasificados con la etiqueta j por los analistas y A_i^j es la predicción del modelo para el $frame\ i-$ésimo considerando solo los $frames$ etiquetados como j por los expertos. Esta fórmula puede ser entendida también como el porcentaje de clasificaciones correctas sin considerar la etiqueta 1 correspondiente a silencio (SIL).

Uno de los objetivos particulares del trabajo es obtener buenos resultados de clasificación de los eventos sísmicos, puesto que la detección de este evento puede tener interesantes aplicaciones prácticas. Por esta razón, se calcula el porcentaje de clasificaciones correctas de la etiqueta ERQ. Para calcular el *accuracy* de esta etiqueta se toman en consideración sólo los $frames$ etiquetados como ERQ por los analistas y se mide el porcentaje de estos $frames$ que fueron correctamente clasificados como ERQ por el modelo entrenado. La fórmula aplicada para el cálculo se muestra a continuación:

$$accuracy\ ERQ = 100 \cdot \frac{\sum_{i=1}^{N_3} [A_i^3 = 3]}{N_3}$$

Donde N_3 es el número total de $frames$ clasificados con la etiqueta 3 (correspondiente a ERQ) por los expertos y A_i^3 es la predicción del modelo para el $frame\ i-$ésimo considerando solo los $frames$ etiquetados como 3 por los analistas.

Las tres métricas calculadas comparan $frame$ a $frame$ las etiquetas correctas con las entregadas por los modelos. Esta forma de calcular los resultados implica que no solo se intenta detectar los eventos, sino que también se busca acertar en el tiempo de inicio y de final de estos. Esto aumenta la dificultad para obtener altos porcentajes de clasificaciones correctas, puesto que exige una mayor precisión a los modelos.

Capítulo 5

Resultados

5.1. Base de Datos

A partir de los pasos explicados en la metodología, se obtienen 1.381 subgrabaciones de aproximadamente 600 $frames$, correspondiente a alrededor de 10 minutos de audio cada una. Se asignan 1.036 grabaciones para entrenar (75 %) y 345 grabaciones para la prueba (25 %) al dividir la base de datos en conjunto de entrenamiento y de *test*. El conjunto de entrenamiento contiene en total 177 horas y 20 minutos de audio, con 30.772 eventos en total, de los que 15.136 (49.2 %) son eventos distintos de SIL. El conjunto de *test* contiene 59 horas y 4 minutos de audio en total, con 10.085 eventos en total, de los que 4.961 (49.2 %) son eventos distintos de silencio.

En la tabla 5.1 se muestra en más detalle la distribución de cada etiqueta en el conjunto de entrenamiento. En la columna $N°$ *eventos* se muestra el número de eventos totales de cada etiqueta y en la columna $N°$ $frames$ se muestra el número de $frames$ totales que pertenecen a cada etiqueta, considerando todas las grabaciones para entrenar. Se puede ver que el evento más numeroso es el silencio, apareciendo en 15.636 ocasiones distintas y abarcando más del 90 % de los $frames$ totales. Otro evento que aparece bastante es *Fin whale song* (FWS) con 11.261 ocurrencias y 13.241 $frames$ asignados. La etiqueta ERQ aparece 233 veces, pero abarca 6.667 $frames$ dada la mayor duración de estos eventos.

Tabla 5.1: Cantidad de eventos y de *frames* de cada etiqueta en el conjunto de entrenamiento

Etiqueta	N° eventos	N° *frames*
SIL	15.636	577.014
UND	52	165
ERQ	233	6.667
S13	38	324
S21	132	1.533
S22	557	4.365
S23	483	3825
SEP	67	612
FWS	11.261	13.241
FWD	563	597
FWD2	160	182
FWD3	94	106

Etiqueta	N° Eventos	N° *frames*
SWD	24	26
SWU	234	297
MI	52	91
SHIP	11	5.561
AA	1.008	7.614
UNDS22	14	19
UNDAA	32	42
ERQFWS	26	31
S22FWS	9	12
S23FWS	19	21
FWSFWD	6	6
AAFWS	70	85

En la tabla 5.2 se muestra en detalle la distribución de cada etiqueta en el conjunto de *test*. La columna *N° eventos* muestra el número de eventos totales de cada etiqueta y la columna *N° frames* muestra el número de *frames* totales asignados a cada etiqueta. El evento más numeroso es el silencio, con 5.124 ocurrencias distintas y con más de 90 % de los *frames* totales asignados. El evento *Fin whale song* (FWS) aparece en numerosas ocasiones con 3.630 apariciones y 4.201 *frames* asignados. La etiqueta ERQ ocurre en 79 ocasiones distintas, abarcando 2.243 *frames*.

Tabla 5.2: Cantidad de eventos y de *frames* de cada etiqueta en el conjunto de prueba

Etiqueta	N° eventos	N° *frames*
SIL	5.124	192.072
UND	21	55
ERQ	79	2.243
S13	13	108
S21	40	514
S22	177	1.469
S23	155	1.287
SEP	24	206
FWS	3.630	4.201
FWD	196	201
FWD2	55	62
FWD3	33	41

Etiqueta	N° eventos	N° *frames*
SWD	5	5
SWU	81	100
MI	30	46
SHIP	4	2.112
AA	354	2.562
UNDS22	4	6
UNDAA	12	14
ERQFWS	10	11
S22FWS	4	4
S23FWS	7	7
FWSFWD	2	2
AAFWS	25	29

5.2. Gaussian Mixture Model

Los resultados obtenidos dependen fuertemente del método usado para representar las grabaciones. Por esta razón, resulta esencial encontrar los parámetros óptimos de los bancos de filtros con los que se obtienen los vectores de *features*.

Como ya se mencionó, son tres los parámetros que definen el banco de filtros: N, f_{th} y α. Para reducir el espacio de búsqueda, se fijó el valor $\alpha = 30°$, puesto que los resultados no cambian en gran medida al modificar este parámetro. El parámetro N, correspondiente al número total de filtros, se fija para tener tres distintas frecuencias de truncado f_m. La máxima frecuencia capturada con los hidrófonos corresponde a $f_{max} = 125Hz$ y se utilizan como f_m los valores: f_{max}, la mitad de f_{max} $(62.5Hz)$ y un cuarto de f_{max} $(31.25Hz)$. La frecuencia de umbral f_{th}, a partir de la cual se aumenta linealmente el ancho de banda de los filtros, se determina a partir de las frecuencias de truncado. Se realizan experimentos usando $f_{th} = f_m$, $f_{th} = \frac{3}{4}f_m$, $f_{th} = \frac{1}{2}f_m$ y $f_{th} = \frac{1}{4}f_m$, o sea, frecuencia de umbral igual, tres cuartos, un medio y un cuarto de la frecuencia de truncado. De esta manera, se pueden comparar de manera directa los resultados obtenidos con distinta frecuencia de corte.

En la tabla 5.3 se muestra el porcentaje de clasificaciones correctas de los experimentos al usar el modelo HMM sin saltos de la figura 4.8. Se puede ver que se obtienen mejores resultados con frecuencia de truncado $f_m = 31.25Hz$, obteniéndose el mayor *accuracy* con frecuencia umbral igual a la de truncado ($f_{th} = 31.25Hz$), o sea, usando filtros de igual ancho de banda.

Tabla 5.3: *Accuracy* de *test* de experimentos con distintas combinaciones de parámetros N y f_{th} en el banco de filtros usado. Experimentos con modelo HMM sin saltos desde el primer estado

	f_{th}			
f_m	f_m	$\frac{3}{4}f_m$	$\frac{1}{2}f_m$	$\frac{1}{4}f_m$
125 Hz	81.3	87.3	86.2	87.1
62.5 Hz	87.7	87.9	88.5	87.5
31.25 Hz	90.2	88.6	86.9	89.9

En la tabla 5.4 se presenta el porcentaje de *recall* de los experimentos de *test* con el modelo HMM sin saltos. Se puede ver que se obtienen porcentajes bajos, siendo ligeramente mejores cuando la frecuencia de truncado usada es $f_m = 125Hz$, o sea, sin truncar el rango de frecuencias.

Tabla 5.4: *Recall* de *test* de experimentos con distintas combinaciones de parámetros N y f_{th} en el banco de filtros usado para obtener $features$. Experimentos con modelo HMM sin saltos desde el primer estado

	f_{th}			
f_m	f_m	$\frac{3}{4}f_m$	$\frac{1}{2}f_m$	$\frac{1}{4}f_m$
125 Hz	41.9	46.5	47.4	44.0
62.5 Hz	43.7	45.6	43.8	23.6
31.25 Hz	43.7	36.3	23.9	12.9

En la tabla 5.5 se puede ver el porcentaje de *accuracy* de *test* para la etiqueta particular ERQ, utilizando el modelo HMM sin saltos. Se obtienen porcentajes de clasificaciones correctas alrededor de 70 %, con un *peak* de 81.5 % al usar frecuencia de truncado $125Hz$ y frecuencia umbral $125Hz$.

Tabla 5.5: *Accuracy* de *test* para etiqueta ERQ. Experimentos con distintas combinaciones de parámetros N y f_{th} en el banco de filtros utilizado. Experimentos usando modelo HMM sin saltos desde el primer estado

	f_{th}			
f_m	f_m	$\frac{3}{4}f_m$	$\frac{1}{2}f_m$	$\frac{1}{4}f_m$
125 Hz	81.5	71.8	76.9	68.5
62.5 Hz	69.6	66.3	67.5	64.5
31.25 Hz	71.5	72.5	73.7	72.3

En la tabla 5.6 se presenta el porcentaje de clasificaciones correctas de los experimentos al usar el modelo HMM con saltos de la figura 4.10. Se puede ver que los resultados son ligeramente inferiores a los obtenidos en los experimentos de la tabla 5.3. Las combinaciones de f_{th} y f_m muestran tendencias similares, se tiene que el mejor resultado es con frecuencia de truncado $f_m = 31.25Hz$ y frecuencia umbral igual a la de truncado.

Tabla 5.6: *Accuracy* de *test* de experimentos con distintas combinaciones de parámetros N y f_{th} en el banco de filtros usado para obtener $features$. Experimentos utilizando modelo HMM con saltos desde el primer estado

	f_{th}			
f_m	f_m	$\frac{3}{4}f_m$	$\frac{1}{2}f_m$	$\frac{1}{4}f_m$
125 Hz	82.2	86.0	85.5	88.3
62.5 Hz	86.4	86.8	86.8	87.6
31.25 Hz	89.8	88.2	87.4	89.4

Los resultados de *recall* de los experimentos de *test*, usando el modelo HMM con salto desde el primer estado, se presentan en la tabla 5.7. Se puede ver que son valores similares a los obtenidos con el modelo HMM sin salto de la tabla 5.4.

Tabla 5.7: *Recall* de *test* de experimentos con distintas combinaciones de parámetros N y f_{th} en el banco de filtros implementado. Experimentos con modelo HMM con saltos desde el primer estado

	f_{th}			
f_m	f_m	$\frac{3}{4}f_m$	$\frac{1}{2}f_m$	$\frac{1}{4}f_m$
125 Hz	39.8	48.1	47.7	43.3
62.5 Hz	47.6	46.0	45.2	25.2
31.25 Hz	41.8	35.4	22.5	13.1

En la tabla 5.8 se muestra el porcentaje de *accuracy* de *test* para la etiqueta ERQ, usando el modelo HMM con saltos. Se obtienen porcentajes de clasificaciones correctas superiores a 70 % en varios experimentos. El mayor porcentaje (74.8 %) se obtiene usando frecuencia de truncado $62.5Hz$ y frecuencia umbral $62.5Hz$.

Tabla 5.8: *Accuracy* de *test* de etiqueta ERQ. Experimentos con distintas combinaciones de parámetros N y f_{th} en el banco de filtros usado. Experimentos usando modelo HMM con saltos desde el primer estado

	f_{th}			
f_m	f_m	$\frac{3}{4}f_m$	$\frac{1}{2}f_m$	$\frac{1}{4}f_m$
125 Hz	74.6	67.8	65.9	72.8
62.5 Hz	74.8	69.8	72.9	70.9
31.25 Hz	67.9	68.8	68.4	68.1

5.3. Deep Neural Network

Para realizar experimentos usando redes neuronales resulta necesario definir dos parámetros fundamentales del modelo: el número de capas de la red (N_{layers}) y el número de neuronas ocultas en las capas intermedias de la red (N_{hidden}). Cabe destacar que se usa el mismo número de neuronas ocultas para todas las capas intermedias. Otros parámetros a determinar corresponden a los tres parámetros que definen el banco de filtros: N, f_{th} y α. Aquí se utiliza la mejor combinación encontrada en los experimentos con GMM, por lo que se usa N de tal manera que la frecuencia de truncado es $31.25Hz$, se usa $f_{th} = f_m = 31.25Hz$ y $\alpha = 30°$. En base a los resultados obtenidos con GMM, se utiliza el modelo HMM sin salto desde el primer estado de la figura 4.10 para realizar la búsqueda de la combinación óptima de N_{layers} y N_{hidden}.

El número de neuronas en las capas ocultas de la red DNN usado concuerda con el número de neuronas de codificación del *autoencoder*. A su vez, el tamaño de la codificación depende del número inicial de $features$, puesto que se usa un número igual o menor para obtener una nueva representación. El número de features es 189 (63 a partir del banco de filtros usado, más sus coeficientes Δ y $\Delta\Delta$ respectivos), por lo que se prueba con 189 (igual al número de $features$), 142 ($\frac{3}{4}$ del número de $features$) y con 95 ($\frac{1}{2}$ del número de $features$). Así, el número de neuronas en las capas ocultas probados son $N_{hidden} = 189$, $N_{hidden} = 142$ y $N_{hidden} = 95$. Se prueba con $N_{layers} = 3$, $N_{layers} = 5$ y $N_{layers} = 7$ para el número de capas de la red neuronal (no se usa un número mayor porque los resultados decrecen drásticamente después de $N_{layers} = 7$).

En la tabla 5.9 se muestran los resultados de clasificaciones correctas sobre el conjunto de *test* con las distintas combinaciones de N_{layers} y N_{hidden} probadas. Se puede ver que se obtienen porcentajes de *accuracy* mayores usando 7 capas para la red neuronal, obteniéndose el mayor porcentaje de clasificaciones correctas (89.08 %) con 95 neuronas ocultas en las capas ocultas.

Tabla 5.9: *Accuracy* de *test* de experimentos con distintas combinaciones de número de capas y de neuronas ocultas para el modelo DNN

	N_{hidden}		
N_{layers}	189	142	95
3	84.1	85.7	83.0
5	85.4	84.5	81.7
7	88.6	87.5	89.1

En la tabla 5.10 se pueden ver los resultados de *recall* de *test* obtenidos probando las distintas combinaciones de N_{layers} y N_{hidden}. Se puede ver que se tienen porcentajes mayores usando 3 capas para la red neuronal, obteniéndose porcentajes superiores a un tercio con los tres valores de N_{hidden} probados.

Tabla 5.10: *Recall* de *test* de experimentos con distintas combinaciones de número de capas y de neuronas ocultas para el modelo DNN

	N_{hidden}		
N_{layers}	189	142	95
3	36.1	34.0	34.9
5	33.4	34.0	30.4
7	28.4	23.8	24.9

Se puede ver a partir de la tabla 5.11 que los mejores resultados de *accuracy* de la etiqueta ERQ se obtienen usando una red con 3 capas ocultas, con porcentajes de clasificaciones correctas muy altos sobre 97 %. El resto de las combinaciones presentan porcentajes de *accuracies* ligeramente menores pero altos de igual forma.

Tabla 5.11: *Accuracy* de *test* de etiqueta ERQ. Experimentos con distintas combinaciones de número de capas y de neuronas ocultas para el modelo DNN

	N_{hidden}		
N_{layers}	189	142	95
3	97.3	97.5	97.7
5	94.6	96.5	93.1
7	94.7	93.3	92.4

Debido a los objetivos del trabajo, se escoge como modelo óptimo el modelo con mayor *accuracy* total, correspondiente a la red DNN de 7 capas ocultas y 95 neuronas en las capas ocultas. En base a este resultado, se procede a probar otras combinaciones de banco de filtros, manteniendo el número de capas ocultas en 7. Se prueba usando como número de neuronas en la capa oculta el mismo número de $features$, puesto que entrega consistentemente buenos resultados (ver primera columna de resultados de las tablas 5.9, 5.10 y 5.11). Se realizan experimentos usando frecuencias de truncado $125Hz$, $62.5Hz$ y $31.25Hz$. Se fija la frecuencia umbral en $f_{th} = 31.25Hz$ para los tres experimentos, basado en los resultados consistentemente buenos con este valor usando GMM.

En el caso con frecuencia de truncado $125Hz$ se tienen $features$ de 219 elementos, por lo que se usa una red DNN con 219 neuronas en las capas ocultas. En el experimento con $f_m = 62.5Hz$ se tienen $features$ de 210 componentes, por lo que se implementa la red DNN con $N_{hidden} = 210$. Finalmente, en el caso con frecuencia $f_m = 31.25Hz$ se tienen $features$ de 189 elementos, por lo que la red DNN usada tiene 189 neuronas en las capas ocultas.

En la tabla 5.12 se muestran los resultados obtenidos probando distintos bancos de filtros para la obtención de los $features$ iniciales. Cabe recordar que en los experimentos de la tabla 5.9 se usó sólo un banco de filtros en particular y se determinó a partir de estos resultados el número de capas ocultas y el número de neuronas ocultas óptimos. Con estos experimentos se exploran distintos bancos de filtros para averiguar si se pueden obtener mejores resultados.

La tabla 5.12 muestra el *accuracy* general, el porcentaje de *recall* y el porcentaje de clasificaciones correctas de la etiqueta ERQ. Se puede ver que el experimento usando frecuencia de truncado $f_m = 125Hz$ entrega el mejor porcentaje de *accuracy* y también el mejor porcentaje de *recall*. Por otra parte, el experimento con frecuencia de truncado $f_m = 62.5Hz$ entrega el mejor porcentaje de *accuracy* del evento sísmico. En base a los resultados se observa que el mejor modelo se obtiene usando $f_m = 125Hz$.

Tabla 5.12: *Accuracy* total, *recall* y *accuracy* de la etiqueta ERQ de experimentos usando tres distintos bancos de filtros. Se usa una red DNN con $N_{layers} = 7$ y número de neuronas ocultas igual al número de $features$ obtenidos a partir de los bancos de filtros.

f_m	Accuracy	Recall	Acc. ERQ
125 Hz	90.1	28.8	91.7
62.5 Hz	84.6	25.5	97.9
31.25 Hz	89.1	24.9	92.4

Capítulo 6

Análisis de Resultados

6.1. Base de Datos

Se puede ver a partir de las tablas 5.1 y 5.2 que el evento silencio abarca consistentemente el 50.8 % de los eventos totales y el 92 % de los $frames$ totales. Los porcentajes calculados en el conjunto de entrenamiento y de *test* coinciden casi exactamente, lo que indica que se tienen suficientes ejemplos para que la ley de los grandes números aplique. La gran diferencia entre el porcentaje de eventos totales y el porcentaje de $frames$ totales de la etiqueta SIL se debe a que los eventos silencio tienen en promedio una mayor duración que la del resto de los eventos. Estos porcentajes muestran la gran presencia de silencio en las grabaciones, resultado esperado dado la naturaleza de las grabaciones.

La razón entre el número de $frames$ y el número de eventos de una etiqueta indica que tan largo es el evento en promedio. Cabe destacar el caso de la etiqueta $SHIP$, donde se tienen eventos de más de 500 $frames$ en promedio (tanto en el conjunto de *test* como en el de entrenamiento). Este resultado muestra porque los analistas utilizaron una ventana de 30 minutos en el espectrograma para detectar este evento en particular. Los eventos sísmicos también destacan por su duración, se puede ver que en promedio tienen una duración de aproximadamente 30 segundos, duración bastante superior a la del resto de los eventos.

La etiqueta FWS (*Fin whale song*) es la etiqueta más numerosa después de SIL, se puede ver que este evento es de corta duración, con duración promedio menor a dos $frames$, lo que significa que los eventos duran menos de dos segundos en promedio. Se observa que la etiqueta $FWSFWD$ presenta pocos eventos para entrenar, por lo que es probable que el modelo tenga problemas para clasificar esta etiqueta. En general, se puede observar que las etiquetas solapadas tienen pocas ocurrencias en la base de datos. Este hecho es de esperar, puesto que estos eventos ocurren en ocasiones especiales donde dos eventos distintos aparecen al mismo tiempo, coincidencia que es poco probable que ocurra. Se puede ver que las etiquetas solapadas no son las únicas con pocos ejemplos, en particular, la etiqueta SWD (*Sei downswep*) presenta pocos eventos y un número de $frames$ totales bajo.

En base a las tablas 5.1 y 5.2 se puede concluir que la gran mayoría de los eventos y $frames$ son silencio, por lo que se tiene un gran desbalance entre clases y, por otra parte, la mayoría de las etiquetas solapadas y algunas etiquetas individuales tienen pocos ejemplos para entrenar. Estos factores hacen que la clasificación correcta de todas las clases sea un proble-

ma difícil de solucionar. Además se observa que varias etiquetas tienen duraciones promedio de pocos segundos, lo que aumenta la dificultad de clasificar correctamente estas etiquetas. Sin embargo, se puede ver que la etiqueta ERQ, correspondiente a los eventos sísmicos, posee una cantidad considerable de eventos y una duración promedio de varios $frames$, lo que indica que probablemente los modelos serán capaces de entrenar y clasificar esta etiqueta.

6.2. Gaussian Mixture Model

A partir de los resultados de la tabla 5.3 se puede observar que no existe una tendencia clara en los resultados al considerar la frecuencia umbral en función de la frecuencia de truncado (comparando columnas de la tabla). Sin embargo, se observa que en el caso con frecuencia de truncado $f_m = 125Hz$ se obtienen buenos resultados usando $f_{th} = \frac{1}{4}f_m = 31.25Hz$, en el caso con $f_m = 62.5Hz$ la mejor combinación es con $f_{th} = \frac{1}{2}f_m = 31.25Hz$, y finalmente cuando $f_m = 31.25Hz$ se obtienen mejores resultados con $f_{th} = f_m = 31.25Hz$. En base a esto, se puede concluir que la frecuencia umbral $f_{th} = 31.25Hz$ es la frecuencia óptima a utilizar. Este resultado indica que la forma del banco no es tan relevante y lo importante es establecer una diferencia en la definición entre las frecuencias bajo y sobre $31.25Hz$. Esto sugiere que la información más relevante para clasificar correctamente la gran mayoría de los $frames$ está bajo los $31.25Hz$.

Los resultados de *recall* de la tabla 5.3 indican que los modelos clasifican correctamente la etiqueta más numerosa correspondiente a silencio (SIL) en detrimento de las etiquetas menos numerosas. Los resultados cercanos a 50 % indican que aproximadamente uno entre dos $frames$ con etiqueta distinta de SIL son clasificados correctamente. Este resultado es muy superior al resultado mínimo 4.34 % correspondiente a tirar una moneda al azar entre las 23 clases restantes, pero no es un porcentaje ideal. Claramente el desbalance inevitable en la cantidad de etiquetas resulta en un modelo entrenado correctamente para detectar silencios, pero con problemas para clasificar el resto de las etiquetas.

La etiqueta ERQ corresponde al evento más importante dados los objetivos definidos en este trabajo. Los resultados de la tabla 5.5 muestran que este evento es clasificado de buena manera por el modelo GMM usando el modelo HMM sin salto. Se tienen porcentajes de clasificaciones correctas notablemente superiores al promedio de las etiquetas menos numerosas, con varios experimentos con porcentajes superiores a 70 %.

Los resultados de *test* obtenidos usando el modelo HMM con salto (tabla 5.6) confirman que el mejor valor de frecuencia umbral es $31.25Hz$. Tanto para el caso con frecuencia de truncado $125Hz$ como para el caso con frecuencia $31.25Hz$ se obtiene el mayor *accuracy* con esta frecuencia umbral. Se observa que la combinación de parámetros que entrega el mayor *accuracy* es $f_m = 31.25Hz$ y $f_{th} = 31.25Hz$, lo que concuerda con lo obtenido con el modelo HMM sin salto.

Los resultados de *recall* usando el modelo HMM con salto son similares a los obtenidos con el modelo sin salto. Se obtienen porcentajes cercanos a un tercio en la mayoría de los experimentos. Los resultados de *accuracy* del evento sismo (ERQ) no presentan una clara diferencia al usar o no usar el salto desde el primer estado. Al utilizar el modelo HMM con

salto se obtienen resultados ligeramente mejores cuando $f_m = 62.5Hz$ y al usar el modelo HMM sin salto se obtienen resultados ligeramente superiores con la frecuencia de truncado $f_m = 125Hz$.

En general, se observa que los resultados de *accuracy* son mejores al usar el modelo HMM sin salto. Los resultados de *recall* y de *accuracy* de la etiqueta ERQ no muestran una tendencia clara dependiendo del modelo usado, pero se observa que el experimento con $f_m = 125Hz$ y $f_{th} = 125Hz$ es consistentemente el mejor para estas métricas. Resulta interesante que este modelo a la vez entrega el peor resultado de *accuracy* total, esto sugiere que en este caso el modelo se entrena mejor para clasificar las etiquetas menos numerosas, pero en consecuencia entrega una peor clasificación para la etiqueta de silencio, lo que resulta en un porcentaje de *accuracy* general inferior.

6.3. Deep Neuronal Network

A partir de los resultados mostrados en la tabla 5.9 se puede ver que los modelos usando red neuronal de 7 capas ocultas entregan mejor porcentaje de *accuracy* de *test*. Por otra parte, no se observa una tendencia clara para el número de neuronas ocultas. Se observa que $N_{hidden} = 189$ es la mejor opción cuando $N_{layers} = 5$, $N_{hidden} = 142$ es la mejor opción cuando $N_{layers} = 3$ y $N_{hidden} = 95$ es la mejor opción cuando $N_{layers} = 7$, por lo que no hay un patrón para este parámetro.

Los resultados de *recall* de la tabla 5.10 indican que se obtiene una mejor clasificación de las etiquetas menos numerosas al usar $N_{layers} = 3$. A la vez, se puede ver que se obtienen porcentajes de *recall* inferiores al usar 7 capas en la red DNN. Se observa que 189 es un buen valor para N_{hidden}, entregando resultados relativamente altos en los tres experimentos. En general, se tiene porcentajes de *recall* cercanos a 33 %, lo que significa que aproximadamente uno entre tres eventos distintos de silencio es clasificado correctamente. La gran discrepancia entre los resultados de *accuracy* y de *recall* indican que los modelos entrenados clasifican correctamente la etiqueta más numerosa SIL, pero no así el resto de las etiquetas. El *recall* obtenido es claramente superior al porcentaje mínimo de 4.34 %, lo que sugiere que el modelo si entrenó las etiquetas menos numerosas, pero no de buena manera debido al gran desbalance de etiquetas.

Los resultados de *accuracy* de la etiqueta ERQ son muy altos. En la tabla 5.11 se puede ver que el porcentaje de clasificaciones correctas es superior al 90 % en todos los experimentos. Claramente los modelos quedan mejor entrenados para clasificar esta etiqueta al usar una red de 3 capas, con porcentajes de *accuracy* muy buenos de 97 % aproximadamente. Los altos resultados de *accuracy* para esta etiqueta muestran que efectivamente se logra entrenar el modelo para algunas de las etiquetas distintas de SIL. Este evento tiene la característica de ser de muy baja frecuencia, esto resulta interesante puesto que los experimentos con GMM indican que la mayor cantidad de información se encuentra bajo los $31.25Hz$, lo que puede explicar los altos resultados obtenidos con esta etiqueta. Otro factor es la duración de estos eventos, que es de varios $frames$, lo que aumenta la probabilidad de ser detectado por el modelo.

Los experimentos de la tabla 5.12 contrastan el mejor modelo de la tabla 5.9 con modelos

usando distintos bancos de filtros para obtener los $features$ iniciales. Se puede ver que los resultados mejoran al usar un banco de filtros que usa todo el rango de frecuencias (experimento con $f_m = 125Hz$, o sea sin truncar). Se puede ver que usando todo el rango de frecuencias se obtiene un *accuracy* superior de 90.14 % y aumenta también el porcentaje de *recall*, por lo que se clasifica mejor las etiquetas menos numerosas.

Los porcentajes de clasificaciones correctas de los eventos sísmicos es superior a 90 % en los tres experimentos, lo que muestra la robustez de los modelos DNN para detectar este evento. Este hecho resulta interesante, puesto que los buenos resultados de la tabla 5.11 se podrían deber a la baja frecuencia de truncado utilizada esos experimentos (debido a que los eventos sísmicos destacan por su baja frecuencia). Sin embargo, los resultados de la tabla 5.12 indican que aun usando todo el rango de frecuencias se obtienen buenos resultados de *accuracy* para la etiqueta ERQ.

6.4. Comparación entre resultados con GMM y con DNN

En términos de clasificaciones correctas totales se puede ver, al comparar el mejor resultado de las tablas 5.3 y 5.6 con el mejor resultado de la tabla 5.9, que los mejores modelos GMM y DNN entregan valores prácticamente iguales de *accuracy*. El *accuracy* máximo usando GMM es 90.2 % y se obtiene con el modelo HMM sin salto, usando un banco de filtros de parámetros $f_m = f_{th} = 31.25Hz$ y $\alpha = 30°$. El *accuracy* máximo usando DNN (con modelo HMM sin salto) de 90.1 %, usando un banco de filtros de parámetros $f_m = 125Hz$, $f_{th} = 31.25Hz$ y $\alpha = 30°$. En base a los parejos resultados se debe recurrir a las otras métricas para determinar el mejor modelo.

Los resultados de *recall* muestran que, considerando las etiquetas distintas de SIL, se obtienen mejores porcentajes de clasificaciones correctas usando modelos GMM. Con GMM se tienen experimentos con *recall* cercanos a 50 %, en cambio los resultados de los experimentos con DNN se mueven alrededor de 33 %. Estos resultados indican que aproximadamente 1 de cada 2 etiquetas distintas de SIL son clasificadas correctamente por los mejores modelos GMM, contra 1 de cada 3 usando modelos DNN.

Los resultados de *accuracy* de la etiqueta correspondiente al evento sismo muestra resultados diferentes. En este caso los modelos DNN entregan porcentajes ampliamente superiores a los modelos GMM. Se puede ver que los modelos GMM logran entrenar bien para detectar esta etiqueta, con porcentajes de clasificaciones correctas sobre 70 % en muchos experimentos, pero los modelos DNN logran resultados muy buenos, sobre 97 % en algunos experimentos. En base a estos resultados, se concluye que los modelos DNN pueden ser utilizados para detectar eventos sísmicos a partir de grabaciones submarinas, lo cual es uno de los objetivos del trabajo.

Los bajos porcentajes de clasificaciones correctas obtenidos con las etiquetas menos numerosas se deben probablemente a la poca cantidad de ejemplos que muchas de las clases disponen. Como se mencionó, varias de las etiquetas corresponden a eventos solapados, los

que ocurren en pocas ocasiones y por ende son difíciles de entrenar para los modelos. Otra posible razón es el gran número de clases usadas para clasificar, lo que disminuye la probabilidad de que los modelos acierten en la etiqueta correcta.

Cabe destacar que la forma de calcular el porcentaje de clasificaciones correctas, considerando cada uno de los $frames$, hace que sea difícil obtener valores altos. Incluso si el modelo detecta cierto evento, pero solo durante la mitad de los $frames$ en los que ocurre, se obtendrá un 50 % de *accuracy* para la etiqueta. Esta forma de calcular los resultados es bastante penalizadora, lo que sin duda afecta los resultados de clasificaciones correctas obtenidos.

En la mayoría de las tablas se observa que los experimentos con mayor porcentaje de *accuracy* a la vez presentan un menor porcentaje de *recall* y de clasificaciones correctas de eventos sísmicos. Esta tendencia sugiere que los modelos con alto porcentaje de *accuracy* total tienden a clasificar los $frames$ como silencio, lo que resulta en un gran número de $frames$ clasificados correctamente, ya que la mayoría de los $frames$ son silencio. A su vez, esto resulta en menores porcentajes de acierto para el resto de las etiquetas, lo que explica la tendencia observada.

Los resultados de clasificación inferiores de las etiquetas menos numerosas usando DNN en comparación a GMM se pueden explicar debido a las características de estos modelos. Las ventajas de las redes neuronales profundas y el *deep learning* aparecen cuando se tienen bases de datos de gran volumen, puesto que estos métodos siguen aumentando su $performance$ a medida que aumenta el tamaño de la base de datos, a diferencia de métodos tradicionales como GMM. En este trabajo en particular no se dispone de una gran base de datos, por lo que no se puede explotar la gran ventaja de las DNN, lo que queda manifiesto también en el bajo número de capas N_{layers} que entregan los mejores resultados de *accuracy*. Claramente la cantidad de datos no es suficiente para entrenar una red de mayor profundidad.

Una de las características de la etiqueta ERQ es su duración de varios $frames$. Es probable que esta característica juegue un rol importante en la correcta clasificación de esta etiqueta, puesto que hace más probable que los modelos la detecten. Un evento de duración de pocos $frames$ es menos probable que sea detectado. La base de datos presenta muchas etiquetas de corta duración, lo que puede explicar la dificultad para detectar y clasificar estas etiquetas. En base a esto, se puede inferir que los modelos DNN clasifican de mejor manera los eventos de larga duración y, en cambio, los modelos GMM logran detectar mejor los eventos de corta duración. Puesto que la mayoría de los eventos distintos de silencio son de poca duración, esto explicaría los mayores porcentajes de *recall* obtenidos con los modelos GMM, así como la mejor clasificación de ERQ usando redes DNN.

Capítulo 7

Conclusión

Se obtienen porcentajes de clasificaciones correctas similares usando modelos GMM y modelos DNN para el cálculo de las probabilidades de observación. El porcentaje de clasificaciones correctas totales obtenido con ambos modelos es 90 % aproximadamente. La diferencia entre ambos modelos radica en el porcentaje de clasificaciones correctas de las etiquetas distintas de silencio y en el porcentaje de *accuracy* de los eventos sísmicos. Se observa que los modelos DNN clasifican de mejor manera la etiqueta ERQ y, por otra parte, los modelos con GMM clasifican mejor las etiquetas distintas de SIL en promedio.

La superioridad de los modelos usando GMM para la clasificación de las etiquetas menos numerosas se debe probablemente a la cantidad de datos disponibles para entrenar los modelos. Se observa que las redes DNN con mejores resultados son de pocas capas, lo que puede indicar que no se dispone de suficientes datos para entrenar correctamente un modelo DNN más profundo. Una de las ventajas de las redes neuronales profundas y el *deep learning* es la capacidad de mejorar los resultados al alimentar el modelo con grandes cantidades de datos, sin estancarse. En este trabajo en particular esta ventaja no se aprovecha, dado que muchas de las etiquetas distintas de silencio disponen de relativamente pocos ejemplos.

Se observa que los modelos aprenden a clasificar el evento silencio de buena manera, lo que se debe a la gran cantidad de ejemplos para esta clase. A la vez, los modelos tienden a clasificar los $frames$ con esta etiqueta (SIL), lo que afecta los resultados de clasificación del resto de las etiquetas.

La gran mayoría de las etiquetas corresponden a silencio, este desbalance es inevitable puesto que en la realidad gran porcentaje de las grabaciones son silencio y ruido de fondo. Modificar la base de datos para reducir el número de $frames$ con silencio podría servir para obtener mejores resultados con el resto de las etiquetas, pero afectaría las capacidades prácticas de los modelos, ya que en la realidad se verán enfrentados a grabaciones con gran número de silencios. Por esta razón, este desbalance de clases es un problema bastante difícil de abordar y que contribuye a los resultados relativamente bajos de *recall* obtenidos.

Los resultados de *recall* indican que los modelos entrenan de manera deficiente la detección de etiquetas poco numerosas. Se pude ver que los porcentajes obtenidos son bastante superiores a realizar una clasificación al azar, lo que muestra que los modelos si entrenan para estas etiquetas, pero claramente el desbalance de clases y el gran número de clases

distintas resulta en porcentajes de clasificaciones correctas bajos para muchas de las etiquetas. Los resultados consistentemente altos de *accuracy* para la etiqueta ERQ confirman que los modelos si logran entrenar correctamente algunas de las etiquetas distintas de silencio.

La forma de calcular los porcentajes de clasificaciones correctas, comparando $frame$ a $frame$ las predicciones del modelo con las etiquetas correctas, dificulta la obtención de porcentajes altos. De esta manera no solo se le pide al modelo detectar los eventos, sino también acertar en el tiempo de inicio y final de estos. Esta demanda de precisión sobre un problema ya complicado permite comprender y apreciar los resultados obtenidos.

Los altos resultados de *accuracy* para el evento silencio y los menores porcentajes de *accuracy* con el resto de las etiquetas muestran la importancia de disponer clases balanceadas para la clasificación. Otro factor que contribuye a los bajos porcentajes de *recall* es la poca cantidad de ejemplos (en términos absolutos, sin comprar porcentualmente con la etiqueta más numerosa) que varías etiquetas tienen en la base de datos, este hecho prueba la dificultad de entrenar correctamente una clase cuando se tienen pocos ejemplos para el proceso de entrenamiento. Se observa que los modelos DNN se ven más afectados por este factor.

En general, se observa una relación inversa entre el porcentaje de *accuracy* general y el porcentaje de *recall* obtenido en los experimentos. Esto indica que los modelos con *accuracy* general alto tienden a clasificar los $frames$ como silencio, lo que resulta en un gran número de aciertos totales, pero un menor número de acierto para las etiquetas distintas de silencio, que son menos numerosas.

Los altos porcentajes de clasificaciones correctas del evento ERQ obtenidos usando DNN para el cálculo de las probabilidades de observación son alentadores. Los modelos con DNN clasificaron correctamente los eventos sísmicos de manera consistente, habilidad que puede ser de mucho uso práctico. Resulta interesante explorar la posibilidad de usar modelos HMM con redes neuronales profundas para la detección y clasificación de movimientos telúricos. En este trabajo se entrenan los modelos para clasificar entre 24 clases y aun así se obtienen *accuracies* sobre 97 % para la etiqueta ERQ en varios experimentos, por lo que un modelo orientado solo a la clasificación de este evento debería entregar aun mejores resultados.

Uno de los objetivos del trabajo es obtener porcentajes de clasificación igual o mejores a los de trabajos similares de clasificación de señales submarinas. El trabajo más parecido y que permite una comparación relativamente directa es el realizado anteriormente en el Laboratorio de Procesamiento y Transmisión de Voz, en el que se obtuvieron *accuracies* de 85 % aproximadamente (Buchan et al. 2019[6]). Se puede concluir que se logró este objetivo particular, pero no se puede asegurar que los resultados sean mejores en general, dado los bajos porcentajes de *recall* obtenidos. Esta métrica no se calcula en el *paper*, por lo que no se puede comparar claramente.

El trabajo busca generar un modelo computacional que permita realizar la clasificación de objetos en señales submarinas sin la necesidad de gran intervención y esfuerzo humano (una vez entrenado el modelo). El modelo logra realizar la clasificación de manera rápida usando técnicas computacionales, lo que ahorra gran cantidad de esfuerzo y tiempo. Dados los resultados obtenidos, la aplicación del modelo para la clasificación de los 24 eventos dis-

tintos depende del uso que se le quiere dar. Claramente algunas etiquetas no son clasificadas correctamente, pero a la vez, los modelos HMM usando DNN clasifican con gran precisión el evento sísmico, por lo que podría ser utilizado para aplicaciones donde este evento es relevante.

Una de las características de este trabajo es el uso de una gran cantidad de clases distintas para la clasificación. Se concluye en base a los resultados que la clasificación correcta de cada clase resulta difícil, en especial cuando se tiene un desbalance de clases tan marcado. Dado el aumento de dificultad que un gran número de clases implica, se hace evidente la necesidad de tener bases de datos con numerosos ejemplos para cada clase. La combinación de gran número de clases y eventos con pocos ejemplos resulta en malos resultados de clasificación para estas etiquetas, como es de esperar.

Otro factor relevante en los resultados de clasificación de una etiqueta resulta ser la duración del evento. Se observa que los eventos de mayor duración, como es el caso de ERQ, son bien detectados y clasificados por los modelos, por otro lado, los modelos presentan dificultades para clasificar los eventos de corta duración (eventos con pocos $frames$ consecutivos en promedio). Los resultados sugieren que los modelos DNN son capaces de entrenar de mejor manera los eventos de mayor duración y, en cambio, los modelos usando GMM presentan mayor capacidad para detectar los eventos más cortos.

En base a las métricas evaluadas se concluye que la elección entre modelos DNN o modelos GMM para el cálculo de las probabilidades de observación del modelo HMM depende del objetivo que se quiere alcanzar. Si se desea detectar y clasificar correctamente eventos sísmicos, entonces resulta mejor optar por un modelo con redes DNN. Si por otra parte el objetivo es clasificar correctamente la mayor cantidad de clases distintas posibles, entonces es preferible usar modelos con mezcla de gaussianas, en especial si se dispone de clases con pocas muestras.

Para trabajos futuros resulta interesante probar los modelos planteados en este trabajo con una base de datos más grande, donde todas las etiquetas dispongan de un número de etiquetas importantes. Teóricamente es probable que los resultados obtenidos con DNN mejoren significativamente, en especial el porcentaje de clasificaciones correctas de las etiquetas distintas de silencio, y que se obtengan resultados superiores a los modelos GMM en las tres métricas calculadas. Es posible que se pueda entrenar correctamente redes DNN de mayor número de capas y así se aprovechen las ventajas de estas redes. Además, de esta manera se podría verificar o refutar la observación de que los modelos con DNN entrenan de mejor manera los eventos de mayor duración y que los modelos con GMM clasifican mejor eventos más cortos, puesto que en este trabajo la poca cantidad de muestras en varias clases no permite asegurar que así sea a partir de los resultados obtenidos.

Bibliografía

[1] K. Stafford, S. Nieukirk, and C. Fox, "Low-frequency whale sounds recorded on hydrophones moored in the eastern tropical pacific," *The Journal of the Acoustical Society of America*, vol. 106, pp. 3687–3698, 12 1999.

[2] D. K. Mellinger and C. W. Clark, "Recognizing transient low-frequency whale sounds by spectrogram correlation," *The Journal of the Acoustical Society of America*, vol. 107, no. 6, pp. 3518–3529, 2000.

[3] F. Samaran, K. M. Stafford, T. A. Branch, J. Gedamke, J.-Y. Royer, R. P. Dziak, and C. Guinet, "Seasonal and geographic variation of southern blue whale subspecies in the indian ocean," *PLOS ONE*, vol. 8, pp. 1–10, 08 2013.

[4] S. J. Buchan, K. M. Stafford, and R. Hucke-Gaete, "Seasonal occurrence of southeast pacific blue whale songs in southern chile and the eastern tropical pacific," *Marine Mammal Science*, vol. 31, no. 2, pp. 440–458, 2015.

[5] M. F. Baumgartner and S. E. Mussoline, "A generalized baleen whale call detection and classification system," *The Journal of the Acoustical Society of America*, vol. 129, no. 5, pp. 2889–2902, 2011.

[6] S. Buchan, R. Mahu, J. Wuth, N. Balcazar-Cabrera, L. Gutiérrez, S. Neira, and N. Yoma, "An unsupervised hidden markov model-based system for the detection and classification of blue whale vocalizations off chile," *Bioacoustics*, vol. 29, pp. 1–28, 01 2019.

[7] H. A. Garcia, T. Couture, A. Galor, J. M. Topple, W. Huang, D. Tiwari, and P. Ratilal, "Comparing performances of five distinct automatic classifiers for fin whale vocalizations in beamformed spectrograms of coherent hydrophone array," *Remote Sensing*, vol. 12, no. 2, 2020.

[8] H. Mohebbi-Kalkhoran, C. Zhu, M. Schinault, and P. Ratilal, "Classifying humpback whale calls to song and non-song vocalizations using bag of words descriptor on acoustic data," pp. 865–870, 12 2019.

[9] F. Pace, F. Benard, H. Glotin, O. Adam, and P. White, "Subunit definition and analysis for humpback whale call classification," *Applied Acoustics*, vol. 71, pp. 1107–1112, 11 2010.

[10] M. Malfante, J. I. Mars, M. Dalla Mura, and C. Gervaise, "Automatic fish sounds classification," *The Journal of the Acoustical Society of America*, vol. 143, no. 5, pp. 2834–2846, 2018.

[11] F. Shabangu, R. Andrew, D. Yemane, and K. Findlay, "Acoustic seasonality, behaviour and detection ranges of antarctic blue and fin whales under different sea ice conditions off antarctica," *Endangered Species Research*, vol. 43, p. 21–37, 09 2020.

[12] B. Moreira, A. Perkusich, and S. Luiz, "An acoustic sensing gesture recognition system design based on a hidden markov model," *Sensors*, vol. 20, p. 4803, 08 2020.

[13] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition*. USA: Columbia University Press, 1990.

[14] S. S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology*, vol. 53, no. 3, pp. 329–353, 1940.

[15] J. Schröder, J. Anemüller, and S. Goetze, "Performance comparison of GMM, HMM and DNN based approaches for acoustic event detection within task 3 of the DCASE 2016 challenge," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pp. 80–84, September 2016.

[16] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 12 1989.

[17] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in *Artificial Neural Networks – ICANN 2006* (S. D. Kollias, A. Stafylopatis, W. Duch, and E. Oja, eds.), (Berlin, Heidelberg), pp. 632–640, Springer Berlin Heidelberg, 2006.

[18] D.-X. Zhou, "Universality of deep convolutional neural networks," *Applied and Computational Harmonic Analysis*, vol. 48, no. 2, pp. 787–794, 2020.

[19] W. C. Cummings and P. O. Thompson, "Underwater sounds from the blue whale, balaenoptera musculus," *The Journal of the Acoustical Society of America*, vol. 50, no. 4B, pp. 1193–1198, 1971.

[20] S. Buchan, R. Hucke-Gaete, L. Rendell, and K. Stafford, "A new song recorded from blue whales in the corcovado gulf, southern chile, and an acoustic link to the eastern tropical pacific," *Endangered Species Research*, vol. 23, no. 3, pp. 241–252, 2014.

[21] M. Saddler, A. Bocconcelli, L. Hickmott, G. Chiang, R. Landea-Briones, P. Bahamonde, G. Howes, P. Segre, and L. Sayigh, "Characterizing chilean blue whale vocalizations with dtags: a test of using tag accelerometers for caller identification," *Journal of Experimental Biology*, vol. 220, pp. 4119–4129, Nov. 2017.

[22] E. Schall, L. Di Iorio, C. Berchok, D. Filun, L. Bedriñana-Romano, S. Buchan, I. Van Opzeeland, R. Sears, and R. Hucke-Gaete, "Visual and passive acoustic observations of blue whale trios from two distinct populations," *Marine Mammal Science*, vol. 36, p. 365–374, 09 2019.

[23] K. Stafford, D. Bohnenstiehl, M. Tolstoy, E. Chapp, D. Mellinger, and S. Moore, "Antarctic-type blue whale calls recorded at low latitudes in the indian and eastern pacific oceans," *Deep Sea Research Part I: Oceanographic Research Papers*, vol. 51, pp. 1337–1346, 10 2004.

[24] B. Miller, E. Miller, S. Calderan, R. Leaper, K. Stafford, A. Širović, S. Rankin, K. Findlay, F. Samaran, I. Van Opzeeland, R. Mccauley, A. Gavrilov, D. Harris, J. Gedamke, E. Bell, and V. Andrews-Goff, "Circumpolar acoustic mapping of endangered southern ocean whales: Voyage report and preliminary results for the 2016/17 antarctic circumnavigation expedition," 05 2017.

[25] W. A. Watkins, P. Tyack, K. E. Moore, and J. E. Bird, "The 20-hz signals of finback wha-

les (balaenoptera physalus)," *The Journal of the Acoustical Society of America*, vol. 82, no. 6, pp. 1901–1912, 1987.

[26] R. Charif, P. Clapham, and C. Clark, "Acoustic detections of singing humpback whales in deep waters off the british isles," *Marine Mammal Science*, vol. 17, pp. 751 – 768, 10 2001.

[27] A. Sirovic, E. Oleson, J. S. Buccowich, A. Rice, and A. R. Bayless, "Fin whale song variability in southern california and the gulf of california," *Scientific Reports*, vol. 7, 2017.

[28] J. Gedamke, "Geographic variation in southern ocean fin whale song," *International Whaling Commission report SC/61/SH16*, pp. 1–8, 01 2009.

[29] H. Ou, W. W. L. Au, S. Van Parijs, E. M. Oleson, and S. Rankin, "Discrimination of frequency-modulated baleen whale downsweep calls with overlapping frequencies," *The Journal of the Acoustical Society of America*, vol. 137, no. 6, pp. 3024–3032, 2015.

[30] W. A. Watkins, K. E. Moore, D. Wartzok, and J. H. Johnson, "Radio tracking of finback (balaenoptera physalus) and humpback (megaptera novaeangliae) whales in prince william sound, alaska," *Deep Sea Research Part A. Oceanographic Research Papers*, vol. 28, no. 6, pp. 577–588, 1981.

[31] S. Calderan, B. Miller, K. Collins, P. Ensor, M. Double, R. Leaper, and J. Barlow, "Low-frequency vocalizations of sei whales (balaenoptera borealis) in the southern ocean," *The Journal of the Acoustical Society of America*, vol. 136, pp. 418–423, 12 2014.

[32] S. Español-Jiménez, P. Bahamonde, G. Chiang, and V. Häussermann, "Discovering sounds in patagonia, characterizing sei whale (balaenoptera borealis) downsweeps in the south-eastern pacific ocean," *Ocean Science Discussions*, pp. 1–21, 07 2018.

[33] S. De Angelis, V. Bass, V. Hards, and G. Ryan, "Seismic characterization of pyroclastic flow activity at soufrière hills volcano, montserrat, 8 january 2007," *Natural hazards and earth system sciences*, vol. 7, pp. 467–472, 07 2007.

[34] C. Fox, H. Matsumoto, and T.-K. Lau, "Monitoring pacific ocean seismicity from an autonomous hydrophone array," *Journal of Geophysical Research*, vol. 106, pp. 4183–4206, 03 2001.

[35] S. Yun, S. Ni, M. Park, and W. S. Lee, "Southeast indian ocean-ridge earthquake sequences from cross-correlation analysis of hydroacoustic data," *Geophysical Journal International*, vol. 179, pp. 401 – 407, 07 2009.

[36] J. Caplan-Auerbach, R. P. Dziak, D. Bohnenstiehl, W. Chadwick, and T. K. Lau, "Hydroacoustic investigation of submarine landslides at west mata volcano, lau basin," *Geophysical Research Letters*, vol. 41, pp. 5927–5934, 2014.

[37] R. Dziak, C. G. Fox, H. Matsumoto, and A. E. Schreiner, "The april 1992 cape mendocino earthquake sequence: Seismo-acoustic analysis utilizing fixed hydrophone arrays," *Marine Geophysical Researches*, vol. 19, pp. 137–162, 1997.

[38] M. McKenna, D. Ross, S. Wiggins, and J. Hildebrand, "Underwater radiated noise from modern commercial ships," *The Journal of the Acoustical Society of America*, vol. 131, pp. 92–103, 01 2012.

[39] L. J.G., "Classification of ships using underwater radiated noise," *Underwater Acoustic Data Processing. NATO ASI Series (Series E: Applied Sciences)*, vol. 161, pp. 591–596, 1989.

[40] Y. Simard, N. Roy, C. Gervaise, and S. Giard, "Analysis and modeling of 255 source levels of merchant ships from an acoustic observatory along st. lawrence seaway," *The Journal of the Acoustical Society of America*, vol. 140, pp. 2002–2018, 09 2016.

[41] J. Stanley, S. Van Parijs, and L. Hatch, "Underwater sound from vessel traffic reduces the effective communication range in atlantic cod and haddock," *Scientific Reports*, vol. 7, 11 2017.

[42] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Vesel, "The kaldi speech recognition toolkit," *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 01 2011.

[43] E. O. Brigham and R. E. Morrow, "The fast fourier transform," *IEEE Spectrum*, vol. 4, no. 12, pp. 63–70, 1967.

[44] P. Podder, T. Khan, M. Khan, and M. Rahman, "Comparative performance analysis of hamming, hanning and blackman window," *International Journal of Computer Applications*, vol. 96, pp. 1–7, 06 2014.

[45] D.-J. Jwo, I.-H. Wu, and Y. Chang, "Windowing design and performance assessment for mitigation of spectrum leakage," *E3S Web of Conferences*, vol. 94, p. 03001, 01 2019.

[46] F.-N. Yuan, L. Zhang, J.-T. Shi, X. Xia, and G. Li, "Theories and applications of auto-encoder neural networks: A literature survey," *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 42, pp. 203–230, 01 2019.

Anexo A

Hidden Markov Model

A.1. Algoritmo *forward*

El problema de evaluación consiste en determinar que modelo es más probable que haya generado una cierta secuencia de observaciones, cuando se dispone de varios modelos HMM distintos. Aquí, se busca el modelo HMM M que maximiza la probabilidad P($O|M$), dada la secuencia de observaciones $O = \{O_1, ..., O_T\}$.

Resulta esencial poder calcular de manera eficiente la probabilidad P($O|M$) dado el modelo oculto M. Una opción es calcular todos los posibles caminos con sus probabilidades y finalmente sumarlas, pero el número de operaciones requeridas es de orden $O(N^T)$, con N el número de estados posibles y T el largo de la secuencia.

Una solución mucho más eficiente se basa en la variable *forward* definida como:

$$\alpha_t(i) = P(O_1, ..., O_t, X_t = S_i|M)$$

.

En palabras, $\alpha_t(i)$ es la probabilidad de haber observado la secuencia O_1, ..., O_t y encontrarse en el estado S_i en el tiempo t, dado el modelo M. Usando esta variable se puede calcular P($O|M$) como:

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

Donde $\alpha_t(i)$ se puede calcular recursivamente a partir de la ecuación:

$$\alpha_t(i) = B[i, O_t] \cdot \sum_{j=1}^{N} \alpha_{t-1}(j) A[j, i]$$

Donde *1* $\leq t \leq$ *T - 1,* $1 \leq j \leq N$. Con los casos base $\alpha_1(j) = \Pi[j] \cdot B[j, O_1]$.

La cantidad de operaciones necesarias para calcular P($O|M$) se reduce significativamente, siendo de orden $O(N^2T)$.

Así, se puede calcular de manera eficiente la probabilidad P($O|M$) teniendo un modelo M, con lo que en la práctica se puede comparar la probabilidad dados distintos modelos y

escoger el que mejor explica la secuencia O de observaciones.

De manera similar a la definición de $\alpha_t(i)$, se puede definir la variable *backward* $\beta_t(i)$, definida como:

$$\beta_t(i) = P(O_{t+1}, ..., O_T, X_t = S_i|M)$$

La variable $\beta_t(i)$ corresponde a la probabilidad de encontrarse en el estado S_i en el tiempo t, sabiendo la secuencia de observaciones futuras O_{t+1}, ..., O_T.

Esta variable auxiliar también puede ser calculada iterativamente, según la fórmula:

$$\beta_t(i) = \sum_{i=j}^{N} \beta_{t+1}(j) A[i,j] B[j, O_{t+1}]$$

Donde *1* $\leq t \leq$ *T - 1, 1* $\leq j \leq$ *N*. Con los casos base $\beta_T(j) = 1$.

La utilidad de esta variable se hace evidente en la explicación del algoritmo de aprendizaje de Baum-Welch.

A.2. Algoritmo de Baum-Welch

Una pregunta fundamental al trabajar con modelos HMM es como encontrar un modelo óptimo M dada una secuencia de observaciones $O = \{O_1, ..., O_T\}$. Este problema es conocido como el problema de aprendizaje, que busca el mejor modelo para explicar la secuencia de observaciones dadas. Generalmente se resuelve este problema usando el algoritmo de Baum-Welch, que entrega un modelo óptimo local.

Este algoritmo se basa en las variables $\alpha_t(i)$ y $\beta_t(i)$ del algoritmo $forward - backward$, donde $\beta_t(i) = P(O_{t+1}, ..., O_T, X_t = S_i|M)$ y $\alpha_t(i) = P(O_1, ..., O_t, X_t = S_i|M)$. En base a estas variables se define una tercera variable auxiliar denominada $\gamma_t(i)$, definida como:

$$\gamma_t(i) = P(X_t = S_i|O_1, ..., O_t, M)$$

.

Esta variable corresponde a la probabilidad de estar en el estado S_i en el tiempo t, dado un modelo M y una secuencia de observaciones $O = \{O_1, ..., O_T\}$. Se puede calcular fácilmente a partir de $\alpha_t(i)$ y $\beta_t(i)$ por medio de la relación:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O_1, ..., O_t|M)} = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \cdot \beta_t(j)}$$

.

Notemos que con la variable $\gamma_t(i)$ es posible tener un estimado de cuantas veces se pasa por el estado particular S_i, sumando a través de todos los posibles tiempos. Así, $\sum_{t=1}^{T-1} \gamma_t(i)$ es el número esperado de visitas al estado S_i.

Con la definición de estas variables se puede obtener directamente una expresión para la

probabilidad $\xi_t(i,j) = P(X_t = S_i, X_{t+1} = S_j | O_1, ..., O_t, M)$, con:

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot A[i,j]B[j,O_{t+1}] \cdot \beta_{t+1}(j)}{P(O_1,...,O_t|M)} = \frac{\alpha_t(i) \cdot A[i,j]B[j,O_{t+1}] \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{i=j}^{N} \alpha_t(i) \cdot A[i,j]B[j,O_{t+1}] \cdot \beta_{t+1}(j)}$$

Notemos que la variable $\xi_t(i,j)$ permite obtener un estimado de cuantas veces se pasa desde el estado particular S_i al estado particular S_j, sumando a través de todos los posibles tiempos. Así, $\sum_{t=1}^{T-1} \xi_t(i,j)$ es el número esperado de transiciones desde el estado S_i al estado S_j.

A partir de las definiciones dadas, es posible actualizar los parámetros del modelo oculto de Markov (Π, A y B), a partir de las expresiones:

- $\hat{\Pi(i)}$ = frecuencia esperada del estado S_i en el tiempo $1 = \gamma_1(i)$
- $A[\hat{i}, j] = \frac{\text{número transiciones de } S_i \text{ a } S_j}{\text{número de veces que se está en } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$
- $b[j,\hat{O_t}] = \frac{\text{número esperado de veces que se está en } S_j \text{ y se observa } O_t}{\text{número esperado de veces que se está en } S_j} = \frac{\sum_{\substack{t=1 \\ s.t\ O_t}}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$

Así, en cada iteración del algoritmo se obtienen nuevos parámetros Π, A y B del modelo, lo que permite entrenarlo a partir de secuencias de observaciones.

Anexo B

Filtro Triangular de Frecuencias

Una de las aplicaciones más relevantes de filtros de frecuencias es la obtención de la representación *Mel-Frecuency Ceptrum* de una señal. Esta representación usa filtros de acuerdo a la escala de Mel para simular la percepción humana no lineal del sonido. Esta transformación tiene muchas aplicaciones, por ejemplo, sirve para representar óptimamente los sonidos al realizar compresión de audio, de forma que se pierda la menor cantidad de información perceptible por humanos.

B.1. Mel Frecuency Ceptrum (MFC)

La representación MFC del sonido está compuesta por un conjunto de coeficientes denominados *Mel-Frequency Cepstral Coefficients* (MFCC). Estos coeficientes comúnmente se calculan a partir de 5 pasos:

- Calcular la transformada de Fourier de la señal original, usando FTT.
- Aplicar un banco de filtros de frecuencias triangulares, siguiendo la distribución de la escala de Mel (ver figura 2.2).
- Tomar el logaritmo de la energía en cada filtro.
- Calcular la transformada de coseno discreta, considerando el logaritmo de la energía en cada filtro como la señal de entrada.
- Los coeficientes resultantes corresponden a los *Mel-Frequency Cepstral Coefficients* que componen la representación MFC de la señal.

Anexo C

Gaussian Mixture Model

Estimar los parámetros óptimos de un modelo GMM, dado un conjunto de datos de entrenamiento, no es un problema trivial. Debido a la no linealidad de las funciones gaussianas, generalmente no se puede obtener una solución analítica directamente. Una solución a este problema es la aplicación del algoritmo de esperanza-maximización (EM) para obtener las medias y varianzas óptimas de las gaussianas, dado un número fijo de gaussianas a considerar.

C.1. Algoritmo de esperanza-maximización (EM)

En general, el algoritmo de esperanza-maximización se utiliza para encontrar estimadores de máxima verosimilitud de parámetros en modelos probabilísticos. El algoritmo parte con una inicialización de parámetros y luego va alternando entre el paso de esperanza (E) y el paso de maximización (M) hasta converger a parámetros óptimos. En el caso de GMM con K gaussianas como modelo en particular y dados los datos de entrenamiento $X = X_1, ..., X_T$, los pasos del algoritmo son los siguientes:

- **Inicialización:**
  - Seleccionar como medias de las gaussianas $(\mu_1, ...\mu_K)$ muestras de los datos elegidas al azar.
  - Calcular las varianzas iniciales de las gaussianas $(\sigma_1, ...\sigma_K)$ como $\sum_{i=1}^{T}(X_i - \bar{X})$. Con $\bar{X}$ el promedio de los datos. Notar que todas las varianzas iniciales son iguales.
  - Iniciar las probabilidades a priori $\Pi_j = P(Y_j)$ como $\frac{1}{K}$. Con $1 \leq j \leq K$.
- **Esperanza:**
  - Computar la probabilidad de pertenencia de cada muestra a cada gaussiana por medio de la expresión: $r_{ij} = P(Y_j|X_i) = \frac{P(Y_j,X_i)}{P(Y_j)} = \frac{P(X_i|Y_j)P(Y_j)}{\sum_{i=j}^{K} P(X_i|Y_j)P(Y_j)} = \frac{N(X_i|\mu_j,\sigma_j)\Pi_j}{\sum_{i=j}^{K} N(X_i|\mu_j,\sigma_j)\Pi_j}$
- **Maximización:** Calcular nuevos valores para Π_j, μ_j y σ_j.
  - $M_j = \sum_{i=1}^{T} r_{ij}$
  - $\Pi_j = \frac{M_j}{T}$
  - $\mu_j = \frac{1}{M_j} \sum_{i=1}^{T} r_{ij} X_i$

- $\sigma_j = \frac{1}{M_j} \sum_{i=1}^{T} (X_i - \mu_j)^T (X_i - \mu_j)$

Se repiten los pasos de esperanza y de maximización hasta que el valor L converge, con:

$$L = log(P(X)) = log(\prod_{i=1}^{T} P(X_i)) = log(\prod_{i=1}^{T} \sum_{i=j}^{K} P(X_i|Y_j)P(Y_j)) = \sum_{i=1}^{T} log(\sum_{i=j}^{K} P(X_i|Y_j)P(Y_j))$$

Al final del proceso se rescatan los vectores de medias $(\mu_1, ...\mu_K)$ y las varianzas $(\sigma_1, ...\sigma_K)$, que definen el modelo GMM entrenado.

Anexo D

Deep Neuronal Networks

Para disponer de una red neuronal que modele correctamente los datos disponibles es necesario realizar un proceso de entrenamiento, de modo que las salidas de la red calcen con las salidas deseadas. Basados en la descripción de una red neuronal (ver figura 2.8) dada por la ecuación $a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)$ el problema de entrenamiento se reduce a encontrar los pesos w_{jk}^l y los *bias* b_j^l adecuados para obtener salidas correctas en la red neuronal. Para encontrar los valores correctos para estos parámetros se utiliza el algoritmo de gradiente descendiente.

D.1. *Backpropagation* y el algoritmo de gradiente descendiente

Generalmente se dispone de datos de entrenamiento donde se conoce la salida deseada para cada dato. A partir de esta información se puede calcular el error entre la salida de la red y la salida correcta para cada dato de entrenamiento. El algoritmo de gradiente descendiente aprovecha esta posibilidad, ajustando los parámetros de la red neuronal para minimizar el error. Un ejemplo de función de error ampliamente utilizada es la función de error cuadrática dada por la ecuación:

$$e = \frac{1}{2N} \sum_{i=1}^{N} |y(X_i) - a^L(X_i)|^2 \tag{D.1}$$

Donde $X = \{X_1, ..., X_N\}$ son los N datos de entrenamiento, con $y(X_i)$ la salida deseada para el dato $i - ésimo$ y $a^L(X_i)$ la salida de la red neuronal al tomar la entrada X_i.

Para modificar los pesos de la red en el entrenamiento, se calcula la derivada parcial del error en función del peso a ajustar y se multiplica por una constante denominada tasa de aprendizaje. Así, cada peso es ajustado en función de cuanta importancia tuvo en el error actual de entrenamiento. En términos matemáticos, se tiene:

$$\hat{w_{jk}^l} = w_{jk}^l - \nu \Delta_{jk}^l$$

$$\Delta_{jk}^l = \frac{\partial e}{\partial w_{jk}^l}$$

Con ν la tasa de aprendizaje considerada y $\hat{w_{jk}^l}$ el nuevo peso ajustado de acuerdo al error de entrenamiento actual. De manera similar para los *bias* se tiene:

$$\hat{b_j^l} = b_j^l - \nu \Delta_j^l$$

$$\Delta_j^l = \frac{\partial e}{\partial b_j^l}$$

Este método de recalcular los parámetros de la red se denomina método del gradiente descendiente y permite encontrar parámetros óptimos asociados a un mínimo local de la función de error. Para aplicar esta técnica de entrenamiento, el problema se traduce a calcular eficientemente las derivadas parciales Δ_{jk}^l y Δ_j^l, para lo que se define la variable auxiliar $\delta_j^l = \frac{\partial e}{\partial z_j^l}$, donde $z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$ y $a_j^l = \sigma(z_j^l)$.

En la capa de salida $l = L$ se tiene:

$$\delta_j^L = \frac{\partial e}{\partial z_j^L} = \sum_k \frac{\partial e}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial e}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial e}{\partial a_j^L} \sigma'(z_j^L)$$

Donde $\frac{\partial e}{\partial a_j^L}$ se puede calcular directamente a partir de la función de error y $\sigma'(z_j^L) = \frac{\partial a_j^L}{\partial z_j^L}$ se calcula directamente a partir de la función de activación $\sigma(\cdot)$ usada en la red.

Para una capa intermedia l, se tiene:

$$\delta_j^l = \frac{\partial e}{\partial z_j^l} = \sum_k \frac{\partial e}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^L} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^L} = \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l)$$

Donde $\sigma'(z_j^l) = \frac{\partial a_j^l}{\partial z_j^l}$ se puede calcular directamente y δ_j^l se puede calcular conociendo los valores δ_j^{l+1} de la capa siguiente.

Ahora basta relacionar las derivadas parciales Δ_{jk}^l y Δ_j^l con las variables auxiliares δ_j^l. Notemos que para Δ_{jk}^l se tiene:

$$\Delta_{jk}^l = \frac{\partial e}{\partial w_{jk}^l} = \sum_m \frac{\partial e}{\partial z_m^l} \frac{\partial z_m^l}{\partial w_{jk}^l} = \frac{\partial e}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \frac{\partial e}{\partial z_j^l} \sigma(z_k^{l-1}) = \delta_j^l a_k^{l-1}$$

Por lo que conociendo δ_j^l se puede calcular directamente la derivada deseada Δ_{jk}^l. Para la derivada Δ_j^l se tiene la relación:

$$\Delta_j^l = \frac{\partial e}{\partial b_j^l} = \sum_m \frac{\partial e}{\partial z_m^l} \frac{\partial z_m^l}{\partial b_j^l} = \frac{\partial e}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} = \frac{\partial e}{\partial z_j^l}$$

.

Así, es posible calcular Δ_{jk}^l y Δ_j^l directamente a partir de las variables auxiliares δ_j^l. A su vez, para calcular las variables δ_j^l se parte calculando directamente δ_j^L a partir del error y luego se va calculando las δ_j^l en las capas inferiores, hasta llegar a la primera capa de la

red. Esta dirección de la propagación del error, desde la capa final hasta la capa inicial, es la razón por la que el método se denomina *backpropagation*.