Reconocimiento de dígitos aislados mediante Alineación Temporal dinámica y Modelos de Markov Ocultos

TESIS

que para obtener el grado de
Maestro en Ciencias
(Matemáticas Aplicadas e Industriales)
presenta

Gregorio Mesinas Cortéz

Director de tesis: Dr. Joaquín Delgado Fernández

UAM-Iztapalapa, 23 de octubre de 2009

RESUMEN

Uno de los problemas que se presentan en los sistemas de reconocimiento de patrones y en este caso el reconocimiento de dígitos fonéticos, es la dificultad de extraer las características fonéticas de una señal proveniente de un micrófono de una manera automática. Los sistemas comerciales son propietarios y normalmente tienen derechos de patente por lo que no es posible analizar con detalle el código fuente de dichos sistemas. Es conveniente contar con un código con el objeto de analizar a detalle los algoritmos matemáticos que se requieren en la extracción de dichas características.

En esta tesis se presenta la programación de dos algoritmos de reconocimiento de dígitos fonéticos con base a la técnica de modelos de Markov ocultos (HMM, *Hidden Markov Models*) y la técnica de alineación temporal dinámica DTW (*Dynamic Time Warping*). Los prototipos de los algoritmos se desarrollan en el lenguaje de alto nivel Mathematica y en el lenguaje también de alto nivel MatLab. La entrada y salida de sonido en formato WAV de Microsoft se hace en bajo nivel utilizando la biblioteca de MatLab. Se utliza un *corpus* (base de datos fonéticos) cuyo varios corpora (elemento del *corpus*) como referencia de los fonemas acústicos de diez dígitos. Cabe mencionar que se utilzó *Prodivoz*, el cual es un proyecto de base de datos fonéticos realizado en la Universidad Nacional del Rosario, Argentina. Por otro lado en este trabajo se diseñaron algunos *corpora* realizados dentro de la UAMI para comprobar el prototipo tema de esta tesis.

Los resultados que se presentan son tablas de las ejecuciones de los programas que reconocen dígitos hablados y aislados utilizando una computadora digital con un micrófono para la extracción de las características fonéticas. También se presentan pantallas que muestran los resultados de forma gráfica. Así mismo se proporciona el código fuente que permite la generación de los resultados.

Contenido

Capítulo 1

Introducción

Partamos de la idea intuitiva siguiente: Se tiene una base de datos, por ejemplo una base de datos de dígitos del 0 al 9, previamente grabados en algún formato conveniente. Los datos están constituidos en clases, i.e., la clase 'n' está representada por todas las grabaciones que corresponden al dígito 'n'. En principio las grabaciones pudieron ser hechas por distintos hablantes, de distinto género y edad, o características de pronunciación personales o regionales; por ello hablamos de "clases". Cada clase consta de cierto número de elementos. Se le pide a una persona que pronuncie un dígito entre 0 y 9; se le graba mediante algún equipo digital. Entonces tenemos la grabación de su voz (del dígito), con el registro de su espectro sonoro, estos datos los podemos considerar como un vector de la pronunciación desconocida, digamos 'u', mientras que nuestras clases: '0', '1',..., '9' las podemos considerar como vectores prototipo. Ahora deberemos usar algún criterio para medir distancias entre vectores de características sonoras, y el dígito desconocido se reconocerá como 'k' si la distancia mínima a cada instancia de las clases se alcanza en un elemento de la clase 'k'.

Este esquema es sin embargo algo más complicado: una instancia la podemos pensar como una serie de tiempo de duración finita que debido a la frecuencia de muestreo finita (típicamente de 8000 a 40000 Hertz) es una serie de tiempo discreto: $\{s_n\}$, $n = 0, 1, 2, \ldots, T-1$ y como la intensidad de la señal también esta cuantizada (típicamente 8 ó 16 bits) la serie toma un conjunto de valores discretos. Por ejemplo, en el formato WAV cuantizado a 8 bits, los valores varian en el rango 0–250[1], de modo que podemos considerar que s_n toma valores continuos (por ejemplo si normalizamos a la unidad, $s_n \in [0, 1]$). Más aún, como muestra la Figura 1.5, que representa la grabación del fonema /a/, la serie dificilmente se puede considerar estacionaria; por ello se utiliza

[1] http://ccrma.stanford.edu/courses/422/projects/WaveFormat/

la Transformada de Fourier de corto tiempo (Short Time Fourier transform) que consiste en segmentar la señal en ventanas en las que las características espectrales estén bien definidas para extraer de su transformada de Fourier un número finito de coeficientes espectrales (en la jerga de sintesis de voz se conocen como coeficientes cepstrales), que representen bien a la excitación tonal. Las ventanas en general deben traslaparse para capturar las características dinámicas de la señal debidas a la no estacionariedad, de modo que es necesario extraer coeficientes adicionales relacionados con la variación temporal de los coeficientes cepstrales entre ventanas consecutivas y con la aceleración. De esta forma, la serie de tiempo se codifica por una secuencia finita de vectores de características espectrales.[6]

$$\{s_n\}_{n=0,\ldots,T-1} \longrightarrow \{\vec{X}_k\}_{k=0,1,\ldots T_X-1} \equiv \widetilde{X}$$

Donde $\vec{X}_i \in \mathbb{R}^d$, d es el número de características espectrales seleccionadas, T_X es el número de ventanas proporcional a la duración de la señal: $T_X \propto T$. Aún cuando sea posible definir una distancia en el espacio de características espectrales $\mathbb{R}^d$, el problema de comparar dos secuencias de vectores

$$\tilde{X} = \{\vec{X}_0, \vec{X}_1, \ldots, \vec{X}_{T_X-1}\} \quad \text{y} \quad \tilde{W} = \{\vec{W}_0, \vec{W}_1, \ldots, \vec{W}_{T_W-1}\},$$

en general de distinta longitud, es intratable de manera directa pues equivale a medir distancias entre trayectorias. En el presente trabajo se aborda el problema de reconocimiento de dígitos aislados mediante dos técnicas: i)la de alineamiento temporal conocida en la literatura como *Dynamic Time Warping* y que en lo sucesivo traducimos como *alineamiento temporal*, y ii) los modelos de Markov ocultos (HMM).

Un corpus está formado por cierto número de archivos .wav de cada uno de los dígitos del 0 al 9. Cada archivo es procesado usando la Transformada de Fourier de tiempo corto, que consiste en subdividir en ventanas la señal y extraer de cada ventana temporal un número finito de características cepstrales. De esta manera el resultado de procesar una instancia de algún dígito es una secuencia de vectores de características cepstrales. Para el reconocimiento de un dígito hablado, se compara la secuencia de vectores de características del dígito hablado con cada una de las instancias del corpus. Se usa el algoritmo DTW para determinar la secuencia más cercana (usando una noción apropiada de distancia cepstral), el dígito que corresponde a la mínima distancia es el dígito reconocido.

En seguida resumimos brevemente el contenido de esta tesis. En el resto de este capítulo de Introducción se presentan las características más importantes del aparato fonador y del oído humanos. Se presenta un modelo simplificado de producción de voz llamado fuente–filtro. En este modelo, la fuente de producción de voz se debe a:

(a) una excitación tonal (*voiced*) caracterizada por una excitación periódica glotal en forma de tren de ondas; (b) una excitación no tonal (*unvoiced*) producida por la turbulencia del aire al constreñirse el tracto bucal. De acuerdo a este modelo la señal que se recibe es una combinación de las excitaciones tonal y no tonal de la que es necesario separar la componente tonal, caracterizada por un espectro con picos bien definidos llamados formantes. Al final de este capítulo se presenta de manera resumida cómo se lleva a cabo el preprocesamiento para obtener a partir de la señal una secuencia de vectores de características cepstrales y cómo se lleva a cabo el reconocimiento de dígitos aislados usando el alineamiento temporal DTW. En el capítulo 2 se presenta en detalle la transformada de Fourier de tiempo corto, la transformación cepstral y el banco de filtros de Mel que permiten calcular los coeficientes cepstrales y la secuencia de vectores de características. En el capítulo 3 se define la distancia pesada y la de Mahalanobis en el espacio $\mathbb{R}^d$ de características cepstrales. En el capítulo 4 se presenta el alineamiento temporal y el algoritmo de programación dinámica que permite encontrar la trayectoria de alineamiento temporal óptima y se define la distancia entre secuencias de vectores como la suma acumulada de distancias entre vectores individuales a lo largo de la trayectoria óptima. El capítulo 5 presenta la implementación de los módulos principales: el de preprocesamiento y el de reconocimiento. Se describen la función y sintaxis de las funciones en Matlab utilizadas en estos módulos. Ahí se describen brevemente los corpora usados para el reconocimiento de dígitos. En el capítulo 6 se ve los HMM. En el capítulo 7 se presentan los resultados obtenidos en el presente trabajo. El capítulo 8 trata de las conclusiones y perspectivas. En el apéndice aparecen tópicos complementarios que no son usados explícitamente en la implementación de la tarea de reconocimiento; aparecen los programas para preprocesamiento y clasificación, y finalmente las corridas de los programas tanto para DTW como para los HMM.

1.1. Características de la voz humana

En esta sección describiremos brevemente la fisiología y anatomía del aparato fonador y del oído con el fin de tener los antecedentes para entender el modelo simplificado de producción de voz y la escala de frecuencias de Mel, relacionada con la respuesta del oído humano a las distintas frecuencias.

Figura 1.1: Diagrama del proceso de la comunicación humana

1.1.1. Componentes del proceso de la comunicación humana

La comunicación es un proceso esencial para hacer transmitir las ideas o pensamientos entre dos o más personas, que dominan o conocen cierto lenguaje (código). En todo proceso de comunicación, se tienen los siguientes componentes (Figura 1.1): **El emisor.** Es la persona que emite el mensaje, a su vez **el receptor.** Es la persona que recibe el mensaje, lo interpreta y lo procesa. Por otro lado **el canal de transmisión.** Es el medio por medio del cual llega el mensaje, puede ser auditivo, visual, etc.; a su vez **el mensaje.** Es el conjunto de ideas, sentimientos, emociones, instrucciones, etc. que se quieren transmitir. Hacemos uso de un **código.**, que es el conjunto de señales, signos, y/o lenguaje que entienden tanto el emisor como el receptor; todo ésto a su vez genera **ruidos, distorsiones e interferencias**, que viene siendo la interrupción a la comunicación. Y los cuales se dividen en: Internos: pueden ser psicológicos, estados de ánimo, cansancio, stress, etc. Externos: alguna falla técnica, problemas de lenguaje, de traducción, tipográfico, etc.

El proceso de la generación del habla comienza cuando el hablante tiene un mensaje en su mente que quiere transmitir a un oyente. El siguiente paso es la codificación del mensaje en una sucesión de fonemas, que corresponde a los sonidos de las palabras; el hablante debe hacer vibrar las cuerdas vocales con el fin de que emitan la señal de sonido audible. Los comandos neuromusculares deben controlar simultáneamente los movimientos articulatorios, incluyendo la mandíbula, los labios, la lengua y el velo del paladar, así como las fosas nasales.

En la comunicación humana la emisión está a cargo del aparato fonador: laringe, faringe, cuerdas bucales, etc.; la recepción por el oído (externo, medio e interno).

1.1.2. El aparato fonador

En el aparato fonador intervienen la boca, la garganta y los pulmones pues todos ellos participan en las distintas emisiones o sonidos del habla (ver Figura 1.2). El sonido producido por las cuerdas vocales es un sonido "en bruto": no se diferencia del que emiten los animales. Este "ruido" al llegar a la boca, es modificado para convertirse en sonido. Esta modificación es lo que llamamos articulación. Cuando se produce un sonido, el aire fluye a través de estos órganos. La articulación del sonido del habla puede ser excitada de tres posibles maneras:

Excitación tonal. Aquí la glotis esta cerrada, pero por la presión del aire, ésta fuerza a que la glotis se abra y se cierre, provocando un tren de pulsos de onda. Así tenemos lo que se conoce como la "frecuencia fundamental"que se encuentra en un rango de 80 a 350 Hz. Los sonidos de las vocales son ejemplos de este tipo, también tenemos la **Excitación no tonal.** La glotis se abre y el aire pasa por la garganta, esto provoca una turbulencia que genera una señal de ruido, y su forma espectral esta determinada por ciertos canales estrechos. Ejemplos de sonidos no tonales son: 't', 'ch','z'.Por último tenemos la **Excitación transitoria** La presión del aire se eleva cuando se cierra la boca o garganta. Por sucesivas aberturas provoca que la presión del aire caiga inmediatamente y así se da el fenómeno de los sonidos explosivos ("plosive burst"), 't', 'f'. En algunos sonidos del habla, estas tres clases de excitación ocurren en combinación.

Los sonidos tonales se pueden modelar por una suma de funciones senoidales en períodos cortos de tiempo, en cambio los no tonales se pueden modelar por ruido. Es relativamente sencillo distinguir un sonido tonal de uno no tonal. Si la señal es $\{s_n\}_{n=0}^{T-1}$ la energía se define como

$$E = \frac{1}{T}\sum_{n=0}^{T-1} |s_n|^2. \tag{1.1}$$

y la tasa de cruces por cero,

$$zcr = \frac{1}{T}\sum_{n=0}^{T-2} \chi(s_n s_{n+1} < 0) \tag{1.2}$$

donde χ es la función indicadora: 1 si el argumento es cierto, 0 en caso contrario. Naturalmente

$$E_{tonal} \gg E_{no\,tonal} \quad \text{y} \quad zcr_{tonal} \ll zcr_{no\,tonal}. \tag{1.3}$$

La forma espectral de la señal del habla, esta determinada por la forma del tracto vocal, es decir el tubo formado por la garganta, la lengua, los dientes y los labios. Cambiando la forma de este tubo y la salida y entrada del aire a través de la nariz y la boca, es como cambiamos la forma espectral de la señal del habla y por lo tanto la emisión de diferentes sonidos.

Figura 1.2: Esquema del aparato fonador

Figura 1.3: El oído

1.1.3. El oído

El oído es el órgano receptor y es el responsable de la audición y del equilibrio; esta formado por el oído externo, medio e interno. En resumen el proceso de la audición se desarrolla así: El oído externo capta las ondas sonoras, que el oído medio se encarga de convertir en energía mecánica. El oído interno convierte la energía mecánica en impulsos nerviosos, que a continuación se trasladan hasta el cerebro.

Oído externo

El oído externo esta formado por lo que se conoce como pabellón de la oreja (la parte externa del oído) y el canal auditivo (conducto auditivo externo). Los sonidos captados por la oreja entran por el canal auditivo y llegan hasta el tímpano, una membrana delgada cubierta de piel que separa el oído externo del oído medio.

Oído medio

El oído medio está formado por el tímpano (membrana del tímpano) y una pequeña cámara llena de aire que contiene una cadena de tres diminutos huesecillos que conectan el tímpano con el oído interno. Los nombres de los huesecillos responden a su forma: el martillo, que está unido al tímpano; el yunque, que conecta el martillo con el estribo, y el estribo, que está unido a la ventana oval localizada en la entrada del oído interno. Las vibraciones del tímpano son amplificadas mecánicamente por los huesecillos y transmitidas a la ventana oval. El oído medio también contiene dos músculos diminutos. El tensor del tímpano, que está unido al martillo, mantiene el tímpano tenso; el músculo estapedio, que está unido al estribo, estabiliza la conexión entre el estribo y la ventana oval. En respuesta al ruido intenso, el músculo estapedio se contrae, dando más rigidez a los huesecillos para que el sonido transmitido sea menos fuerte. Esta respuesta, llamada reflejo acústico, ayuda a proteger al delicado oído medio del daño que le puede provocar el ruido. La trompa de Eustaquio, un pequeño tubo que conecta el oído medio con la parte posterior de la nariz, permite que el aire del exterior entre en el oído medio. Este tubo, que se abre al tragar, ayuda a mantener una misma presión de aire a ambos lados del tímpano, un factor importante para oír con normalidad y no sentir molestias.

Oído interno

El oído interno es una compleja estructura que consta de 2 partes: la cóclea, el órgano de la audición, y los canales semicirculares, el órgano del equilibrio. La cóclea, un tubo hueco en espiral con forma de caracol, contiene un líquido espeso y el órgano de Corti, formado por miles de diminutas células peludas con pequeñas prolongaciones similares a pelos que se extienden hasta el líquido. Las vibraciones sonoras transmitidas desde los huesecillos del oído medio a la ventana oval del oído interno hacen que vibren el fluido y las proyecciones (filamentos semejantes a las pestañas). En seguida se escucha la señal acústica a lo largo de la membrana basilar. Las diferentes células peludas responden a distintas frecuencias del sonido y las convierten en impulsos nerviosos, es decir que un proceso de traducción neural convierte la señal del espectro a la salida en la membrana basilar en las señales activas en el nervio auditivo, correspondiendo al proceso de extracción de características. En una manera que aún no esta bien entendida, la actividad neural a lo largo del nervio auditivo es convertida en un lenguaje de código a los centros altos de procesamiento con el cerebro, esto es, los impulsos nerviosos son transmitidos a través de las fibras del nervio auditivo, que las transportan hasta el cerebro y finalmente la comprensión del mensaje es archivado.[2]

1.2. Modelo de producción de voz

La producción del habla se puede separar en dos partes: la producción de la señal de excitación y la deformación espectral. En la Figura 1.4 se presenta un modelo simple de la producción del habla. La excitación de la voz es modelada por un generador de un tren de pulsos con un espectro $P(f)$. La excitación no tonal es modelada por un generador de ruido blanco con espectro $N(f)$. La amplitud de la señal del generador de pulsos v y del generador de ruido u es controlada por el hablante. Las excitaciones tonales y no tonales se superponen alimentando al modelo de caja del tracto vocal con la función de transferencia $H(f)$. La función de transferencia $R(f)$ modela la emisión característica de los labios. Entonces el espectro de la señal del habla $S(f)$ esta dado por:

$$X(f) = vP(f) + uN(f) \tag{1.4}$$

$$S(f) = X(f)H(f)R(f). \tag{1.5}$$

La Figura 1.5 muestra la señal del fonema /a/ con una duración de 309 ms. La frecuencia de muestreo es 8000 Hz. La escala en el eje horizontal es en unidades del intervalo de muestreo $\Delta t = 1/fs = .125$ ms.

Figura 1.4: Modelo de producción de voz

Figura 1.5: El fonema /a/ a partir del archivo en formato WAV

Debido a que la serie de tiempo de la señal es no estacionaria, se toman muestras de ventanas temporales por ejemplo cada 10 ms. En la Figura 1.6 se muestran diversas ventanas temporales del fonema /a/. En las cuatro ventanas el patrón parece ser estacionario, en cambio en el fonema completo de la Figura 1.5 es claro que el patrón es no estacionario. La componente de rápida variación en el espectro de potencia son causados por la señal no tonal. La componente de baja frecuencia indicado por la curva continua se conoce como "suavizamiento cepstral" y resulta de eliminar la componente no tonal. Justifiquemos: De acuerdo a (1.5) queremos extraer la componente del espectro $H(f)$ que es la modulación del tracto vocal. Escribamos $S(f) = H(f)U(f)$, donde $U(f) = X(f)R(f)$. Tomando el módulo al cuadrado y logaritmo se tiene

$$\log|S(f)|^2 = \log|H(f)|^2 + \log|U(f)|^2$$

Lo cual nos da una expresión para el espectro de potencia continuo $\log|S(f)|^2$, como la superposición de un espectro de lenta variación en frecuencia $\log|H(f)|^2$ y otro de rápida variación $\log|U(f)|^2$ (no podemos decir de alta frecuencia ya que la variable independiente es frecuencia f, no tiempo). La componente de rápida variación se puede eliminar si se toma la transformada inversa de Fourier y se desechan los coeficientes de orden alto. Debido a la simetría del espectro $|S(f)|^2$, se puede tomar la transformada inversa en coseno. Si las frecuencias son discretas, entonces

$$V(n) = S(f_n), \qquad f_n = n\Delta f$$

con $\Delta f = \frac{1}{N\Delta t}$ el intervalo de frecuencia de muestreo. La transformada inversa discreta de Fourier en cosenos es

$$s(d) = \frac{1}{N}\sum \log|V(n)|^2 \cos\left(\frac{\pi d(2n+1)}{N}\right) \tag{1.6}$$

Se puede entonces filtrar las componentes de rápida variación haciendo un corte para componentes altas de $s(d)$ y volviendo a la escala de frecuencias.

1.3. Preprocesamiento

Hemos dicho que la señal de voz debe subdividirse en ventanas, recorriendo éstas por un offset generalmente menor que el ancho de la ventana, de modo que ventanas

Figura 1.6: Tramos de voz del fonema /a/ subdivididas en ventanas con la función de Hamming en tramos de 25 ms y offset de 10 ms (izq.); las unidades en el eje horizontal de las figuras en la columna izquierda son múltiplos de $\Delta t = 0.125$ ms, que es el recíproco de la frecuencia de muestreo $f_s = 8000$ Hz, los valores en el eje vertical son números en el rango $[-32768, 32767]$ del formato *.wav a 16 bits. En las figuras en la columna derecha se grafican los módulos de los coeficientes de Fourier en escala logaritmica vs. la frecuencia. Más precisamente; en el eje vertical se grafica $10 \log |V(f_n)|^2$ vs. f_n, donde $f_n = n\Delta f$; $\Delta f = 1/(L\Delta t)$ con $L = 200$ la longitud de la señal correspondiente a 25 ms.

consecutivas se traslapan. De cada ventana se extraen un número finito de coeficientes característicos de la señal llamados *coeficientes cepstrales*, $\{c_n\}_{n=0}^{Q-1}$; naturalmente la energía de la señal en la ventana, tal como se definió en (1.1), o su logaritmo, deberá agregarse al vector de características. La variación de los coeficientes ceptrales de una ventana a otra, se describen por la variación de los coeficientes cepstrales respecto de ventanas de unas pocas unidades de offset hacia adelante o hacia atrás, dando así los coeficientes Δc_n y $\Delta^2 c_n$. El total de características de la señal en la ventana i se codifica entonces en el vector de características

$$\vec{X}_i = \{c_0, c_1, \ldots, c_{Q-1}, \Delta c_0, \Delta c_1, \ldots, \Delta c_{Q-1}, \Delta^2 c_0, \Delta^2 c_1, \ldots, \Delta^2 c_{Q-1}, \log |P|\} \quad (1.7)$$

de dimensión $3Q + 1$. Típicamente $Q = 12$, de forma que $\vec{X}_i \in \mathbb{R}^{37}$.

El proceso anteriormente descrito constituye un módulo en el sistema de reconocimiento de voz, que llamaremos de *preprocesamiento*, esquemáticamente se muestra en la Figura 1.7.

1.4. Clasificación basada en patrones

En esta metodología un patrón de prueba consiste de una secuencia de vectores de propiedades espectrales, $\tilde{X} = \{\vec{X}_0, \vec{X}_1, \ldots, \vec{X}_{T-1}\}$, donde $\vec{X}_i$ es un vector espectral de la ventana temporal i y T_X es el número total de ventanas. Se dispone de un conjunto de clases ω_ν, $\nu = 1, 2, \ldots, V$. Cada clase ω_ν representada por K_{ω_ν} secuencias de vectores de propiedades espectrales $\tilde{W}_k = \tilde{W}_{k,\omega_\nu}$, $k = 0, 1, 2, \ldots, K_{\omega_\nu} - 1$. De-

Figura 1.7: Módulo de preprocesamiento

berá compararse la secuencia desconocida $\tilde{X}$ con una secuencia patrón $\tilde{W}$, en general de longitudes T_X, T_W distintas.

Si $D(\tilde{X}, \tilde{W})$ denota cierta distancia entre las secuencias, la clasificación se hará de acuerdo al criterio

$$\tilde{W}_{k^*} = \arg \min_{\tilde{W}_k} D(\tilde{X}, \tilde{W}_k) \tag{1.8}$$

y si $\tilde{W}_{k^*} = \tilde{W}_{k^*,\omega_\nu}$ entonces $\tilde{X}$ pertenece a la clase ω_ν.

Dos problemas se presentan en la tarea de clasificación de acuerdo al criterio (1.8); por un lado las secuencias $D(\tilde{X}$ y $\tilde{W})$ no tienen necesariamente las mismas longitudes T_X, T_W, debido a que el número de ventanas no son necesariamente iguales, más aún, debido a que diferentes hablantes pronuncian un mismo fonema con distinta duración, o aún el mismo hablante puede pronunicar el mismo fonema de manera distinta en distintas instancias: si está cansado o relajado, en situación normal o de estrés, etc. El otro problema, más básico aún es que si logramos alinear bajo algún criterio las ventanas temporales, digamos $\vec{X}_i$ con $\vec{W}_j$, es necesario definir una medida de disimilaridad espectral $d(\vec{X}, \vec{W})$ entre vectores de características del tipo (1.7).

Para resolver los puntos anteriores se introduce una distancia en $\mathbb{R}^d$, siendo d el número de características escogidos para caracterizar completamente una ventana temporal cualquiera. Esta distancia deberá ser capaz de cuantificar la disimilaridad de los espectros de potencia como los mostrados en la Figura 1.6 y los parámetros seleccionados deberán tener propiedades estadísticas deseables, como por ejemplo que estén mínimamente correlacionados, con el fin de no incluir información redundante. A este respecto la literatura menciona dos tipos de coeficientes que se describen brevemente:

LPC Coeficientes de predicción lineal (*linear prediction coding*). En este método la serie de tiempo ventaneada s_t se codifica ajustando un modelo AR(Q) de la forma

$$s_t = a_1 s_{t-1} + a_2 s_{t-2} + \cdots + a_Q s_{t-Q} + \epsilon_t$$

con $\epsilon \sim N(0, \sigma)$, el vector de propiedades espectrales se toma entonces como los coeficientes de regresión a_i, $i = 1, 2, \ldots, Q$.

MFCC Coeficientes cepstrales en la frecuencia de Mel (*Mel frequency cepstral coefficients*). En este método se siguen las ideas expuestas en la sección 2.2, en la que según explicamos, se trata de extraer del espectro de potencias la información contenida en los primeros formantes, eliminando del espectro la componente de rápida variación en frecuencia. Previamente a tomar la transformada de Fourier inversa en cosenos (1.6) el intervalo de frecuencias máximo $[0, f_s]$, donde f_s es la

frecuencia de muestreo, se subdivide en ventanas usando un banco de funciones triangulares uniformemente distribuido en una escala de frecuencias llamada escala de Mel. El espectro ya subdividido y transformado a la escala de Mel es

$$G(k) = \sum_{n=0}^{N/2} \eta_{kn} |V(n)|^2, k = 0, 1, \ldots, \mathcal{K} - 1$$

donde $\mathcal{K}$ es el *número de canales*, un parámetro a seleccionar (típicamente $\mathcal{K} = 22$). Los coeficientes cepstrales de Mel se obtienen entonces tomando la transformada inversa de Fourier en cosenos del espectro en la escala de Mel, similarmente a (1.6)

$$c(q) = \frac{1}{\mathcal{K}} \sum_{k=0}^{\mathcal{K}} G(k) \cos\left(\frac{\pi q(2k+1)}{\mathcal{K}}\right) \tag{1.9}$$

Existe una relación recursiva simple entre los coeficientes LPC y MFCC

$$c(0) = \ln \sigma^2 \tag{1.10}$$

$$c(m) = a_m + \sum_{k=1}^{m-1} c_k a_{m-k}, \quad 1 \leq m \leq Q \tag{1.11}$$

$$c(m) = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \quad m > Q \tag{1.12}$$

La Escala de Mel, propuesta por Stevens, Volkman y Newmann en 1937 [14], es una escala musical perceptual del tonos equiespaciados a juicio de oyentes. El punto de referencia entre esta escala y la frecuencia normal se define equiparando un tono de 1000 Hz, 40 dBs por encima del umbral de audición del oyente, con un tono de 1000 mels. Por encima de 500 Hz, los intervalos de frecuencia espaciados exponencialmente son percibidos como si estuvieran espaciados linealmente. En consecuencia, cuatro octavas en la escala de Hertz por encima de 500 Hz se comprimen a alrededor de dos octavas en la escala mel. El término *mel* proviene de la palabra *melodía* para indicar que la escala se basa en comparación de tonos (*Wikipedia*). La escala de Mel se construye entonces como sigue:

1. Escoja la frecuencia de referencia de 1000 Hz y desígnela como “1000 Mels”.

Figura 1.8: Matriz de correlación de características cepstrales de 750 muestras del dígito 0 (izq.). Lugares en la matriz de correlación con valores mayores que 0.8 en valor absoluto

2. Se les presenta una señal a oyentes y se les pregunta que cambien su escala de frecuencia hasta que el tono que perciban sea el doble del tono de referencia, 10 veces más, etc.; luego se les pide que digan cuando se percibe 1/10 del tono de referencia, 1/100 etc.

3. A partir de estos datos se construye la escala de Mel.

La escala de Mel es aproximadamente lineal debajo de 1KHz y logarítmica por encima. Una fórmula cerrada que se usa con frecuencia (y de hecho se toma como la definición de la escala de Mel) es

$$f_{mel} = 2595 \log\left(1 + \frac{f}{700Hz}\right). \tag{1.13}$$

Los coeficientes LPC fueron usados durante mucho tiempo en sistemas de reconocimiento de voz y fué uno de los mayores logros en su época, paulatinamente su uso fué sustituído por los MFCC, debido principalmente a su propiedad de estar decorrelacionados [10]. El gráfico 1.8 muestra la matriz de correlación de 750 muestras del dígito '0' del corpus E' (véase sección 8.5). Nótese las correlaciones muy bajas fuera de la diagonal principal.

Para alinear temporalmente distintas secuencias de vectores se usa el procedimiento de programación dinámica llamado alineación temporal (*Dynamic Time Warping* o DTW). Las secuencias de vectores tales como $\tilde{X} = \{\vec{X}_0, \vec{X}_1, \ldots, \vec{X}_{T_X-1}\}$ se pueden pensar como trayectorias discretas en el espacio métrico de características cepstrales. Si el inicio de la señal de voz está bien definida, se puede comparar $\vec{X}_0$ con $\vec{W}_0$ con la distancia espectral $d(\vec{X}_0, \vec{W}_0)$. Una alineación temporal de las secuencias es una curva en el producto cartesiano de índices: $P(t) = (i(t), j(t))$, $0 \leq i(t) \leq T_X - 1$, $0 \leq j(t) \leq T_W - 1$, para $1 \leq t \leq T = \text{mín}(T_X, T_W)$, con $i(t)$, $j(t)$ monótonas. Dada una alineación temporal P se puede asignar una distancia entre secuencias

$$D_P(\tilde{X}, \tilde{W}) = \sum_{i=0}^{T-1} d(\vec{X}_{i(t)}, \vec{W}_{j(t)})$$

Figura 1.9: Esquema de un módulo de reconocimiento de voz basado en comparación de patrones

y por tanto una distancia máxima

$$D(\tilde{X}, \tilde{W}) = \max_{P} D_P(\tilde{X}, \tilde{W}) \tag{1.14}$$

donde P corre sobre el conjunto de alineaciones temporales monótonas. Naturalmente el cálculo de la distancia directamente de la definición (1.14) es prohibitivo, dado que el conjunto de alineaciones temporales tiene un conjunto de elementos del orden de N^T, siendo N el número de ventanas ; por ello se recurre al principio de Bellman [1]de programación dinámica en la que a partir de la optimización local de trayectorias y un proceso de retropropagación se obtiene una trayectoria óptima P^* tal que

$$D(\tilde{X}, \tilde{W})_{P^*} = \max_{P} D_P(\tilde{X}, \tilde{W}).$$

La Figura 1.9 muestra la estructura del módulo de reconocimiento basado en reconocimiento de patrones: Una base de datos patrón consiste en una colección de instancias de las distintas clases, un dígito desconocido genera una secuencia de vectores $\tilde{X}$ a la salida del módulo de preprocesamiento. Se busca la distancia de la secuencia $\tilde{X}$ a cada instancia de la base de datos. Aquél patrón que se halle a la mínima distancia decidirá la clase del dígito desconocido.

Capítulo 2

Transformada de Fourier de tiempo corto (STFT)

Típicamente los parámetros espectrales del habla son estimados en intervalos de 20–25ms, una frecuencia de muestreo entre 8000 y 22050 hz., y 8 ó 16 bits de cuantización de la amplitud de la señal. La voz a través de líneas telefónicas, por ejemplo, se transmite a 8000 Hz. Aparatos de grabación como dispositivos MP3 o micrófonos de escritorio manejan 8000 o 16000 Hz. En lo sucesivo denotaremos la señal como

$$s(k) = s(k\Delta t)$$

donde $\Delta t = 1/f_s$ es el intervalo de muestreo.

La envolvente del espectro de potencia típicamente muestra un decaimiento de -6dB[1] por octava[2] La Figura 2.1 muestra un espectro de potencia típico junto con una recta ajustada que ilustra este fenómeno. En efecto, la recta ajustada es

$$y = 64.1377 - 0.577827 f_n$$

donde $y = 10\log_{10}(|V(f_n)|^2)$ de modo que por ejemplo,

20 es 1 octava mayor que 10	$\frac{y(20)-y(10)}{1}$	$=$	-5.77827
40 es 4 octavas mayor que 10	$\frac{y(40)-y(10)}{4}$	$=$	-4.3337
45 es 3 octavas mayor que 15	$\frac{y(45)-y(15)}{3}$	$=$	-5.77827
etc.			

[1]Una cantidad P en decibelios se define como $10\log_{10}(P/P_*)$ donde P_* es una cantidad de referencia. En acústica se acostumbra tomar como nivel de referencia el umbral de percepción de un humano promedio es decir 1 KHz. En nuestro caso $10\log|V(n)|^2 = 20\log|V(n)|$.

[2]Dos frecuencias difieren en una octava si una es el doble de la otra

Figura 2.1: Decaimiento experimental de -6 Db (aproximadamente) por octava. La recta ajustada es $64.1377 - 0.577827f$, frecuencia en KHz (ver texto donde se hace referencia a esta figura).

Se usa un filtro de pre-énfasis para eliminar este efecto

$$\hat{s}(k) = s(k) - 0 \cdot 97s(k-1). \tag{2.1}$$

Una de las técnicas para estudiar señales que no son estacionarias en el tiempo se conoce como la transformada de Fourier de tiempo corto. En esta técnica, se toma un tramo corto de la señal $\hat{s}(k)$, $k = m, m+1, \ldots, m+N-1$ y se hace una subdivision en ventanas, es decir se multiplica por una función de ventaneo que enfatiza al tramo central del tramo de la señal y suaviza las discontinuidades en los extremos de la ventana. en este trabajo usaremos la ventana de Hamming

$$w(k) = 0.54 - 0.46\cos\left(\frac{2\pi k}{N-1}\right) \tag{2.2}$$

y se define como cero fuera del rango $k = 0, 1, \ldots, N-1$. La señal ventaneada es

$$v_m(k) = \hat{s}(k)w(k-m), \quad k = m, m+1, \ldots, m+N-1 \tag{2.3}$$

definiéndose como cero fuera del rango prescrito.

En lo sucesivo se considerara la señal subdividida en ventanas $v_m(k)$ en una ventana particular y por lo tanto abreviaremos por $v(k)$.

La transformada discreta de Fourier de la señal $v(k)$, $k = 0, 1, \ldots, N-1$ es

$$\hat{V}(n) = \sum_{k=0}^{N-1} v(k)e^{-j2\pi kn/N}, \quad n = 0, 1, 2, \ldots, N-1 \tag{2.4}$$

de su definición, se ve que $\hat{V}$ admite una extensión a todo $n \in \mathbb{Z}$, periódica de período $2\pi/N$. Los coeficientes de la TDF forman una sucesión de números complejos, el cuadrado de su módulo se llama el espectro de potencia $|V(n)|^2$. Como los coeficientes de la señal $v(k)$ son reales, se sigue que

$$|V(-n)|^2 = |V(n)|^2 \tag{2.5}$$

por lo tanto basta considerar el espectro de potencia en el rango $n = 0, 1, \ldots, [N/2]$

2.1. La transformación cepstral

De acuerdo a nuestro modelo de voz, el espectro de la señal de voz es resultado del la multiplicación de: (a) la señal de excitación tonal y no tonal $X(f)$ (voiced and unvoiced), (b) la función de transferencia del tracto vocal (laringe y boca), $H(f)$; (c) la función de transferencia de los labios $R(f)$. El efecto neto de la excitación tonal y no tonal $X(f)$ y de los labios $R(f)$ se combina en una sola $U(f)$, así

$$S(f) = X(f)H(f)R(f) = H(f)U(f).$$

Para el reconocimiento de voz, es necesario separar la función de transferencia del tracto vocal $H(f)$ de $U(f)$, que en el espectro se observa como ondeletas de rápida variación superpuestas a una tendencia promedio (sería incorrecto llamarlas de alta frecuencia, ya que el espectro vive en el espacio de frecuencias). La *transformación cepstral* es una estimación de la función de transferencia $H(f)$ eliminando estas ondeletas de rápida variación.

La transformación cepstral parte del espectro de potencia (2.4); luego se toma la transformada de Fourier inversa del *espectro de potencia*:

$$\hat{s}(d) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \log(|V(n)|^2) e^{j2\pi dn/N}, \quad d = 0, 1, 2, \ldots, N-1.$$

El índice d se llama *quefrencia* y $\hat{s}$ o su extensión par $s(d)$ se llama el *cepstrum*. La nomenclatura süi géneris, viene del hecho de que d sería el tiempo si tomásemos la transformada de Fourier de $v(k)$ y correspondientemente el espectro. Los picos más altos en el cepstrum corresponden a los primeros formantes del espectro de potencia. Una oscilación en frecuencia en el espectro de potencia de período $p_f = \nu\Delta f$ corresponde a

$$p_f = \nu\Delta f = \nu\frac{f_s}{N} = \frac{\nu}{N\Delta t}$$

y corresponderá en el dominio cepstral a un pico al tiempo

$$t = \frac{1}{p_f} = \frac{N}{\nu}\frac{1}{f_s} = \frac{N\Delta t}{\nu}$$

y corresponderá a una quefrencia de

$$d = \frac{t}{\Delta t} = \frac{N\Delta t}{\nu\Delta t} = \frac{N}{\nu}.$$

Figura 2.2: El cepstrum: es el módulo de la transformada inversa de Fourier del logaritmo del módulo de la transformada de Fourier de la señal. El suavizamiento cepstral consiste en hacer cero los coeficientes a partir de un índice (marcado por una recta vertical que corresponde al índice 32) para después tomar la transformada de Fourier. El espectro así obtenido es el que se muestra en la Figura 2.3 en línea gruesa

Figura 2.3: (a) Espectro de potencia suavizado marcado en línea gruesa. (b) Amplificación. Las unidades en la vertical son $\log_e |V(f_n)|$; en la horizontal n.

Por lo tanto las oscilaciones rápidas en el espectro corresponden a valores pequeños de ν (corto período) y por lo tanto valores de la quefrencia (d) grandes. Por lo tanto una manera de eliminar estas oscilaciones rápidas es haciendo cero los coeficientes de orden alto de $s(d)$ y aplicando la transformada de Fourier nuevamente. En la Figura 2.2 se muestra la transformada de Fourier $s(d)$; la línea vertical indica el índice $d = 32$, a partir del cual se hacen cero sus coeficientes. En la Figura 2.3 se muestra el resultado de suavizar el espectro de potencia.

2.2. Coeficientes cepstrales de la frecuencia de Mel (MFCC)

La respuesta del oído humano a las distintas frecuencias del espectro se representa mejor por la escala de Mel:

$$f_{mel} = 2595 \log_{10}\left(1 + \frac{f}{700Hz}\right)$$

la cual se muestra en la Gráfica 2.4 Para reflejar esto, el espectro de frecuencia se subdivide en ventanas con un banco de funciones triangulares llamadas banco de filtros de Mel (Mel filter bank), que se definen como sigue: Sea $[0, f_{Nyq}]$ el rango de frecuencias máximo a considerar, donde

$$f_{Nyq} = \frac{f_s}{2},$$

Figura 2.4: Escala de mel

donde f_s es frecuencia de muestreo, se conoce como la frecuencia de Nyquist. La frecuencia de Nyquist es la máxima resolución en frecuencia de la señal, de acuerdo al teorema de muestreo de Shannon. Las componentes de la señal de mayores frecuencias que f_{Nyq} se pueden considerar como ruido.

La frecuencia mínima, depende de la calidad de la señal; por ejemplo, para teléfonos celulares actuales 800 Hz es un valor razonable. Supondremos, para simplificar que la frecuencia mínima es cero.

Para representar la menor respuesta del oído a las altas frecuencias (por ejemplo el oído humano es incapaz de detectar sonidos que un perro sí puede oir), se selecciona un número $\mathcal{K}$ de filtros llamados canales; estos no son más que funciones de ventaneo en forma triangular que se definen como se muestra en seguida; para evitar confusión en lo que resta de esta sección denotaremos frecuencias en la escala de mel y en la escala normal (Hz) por ϕ y f, respectivamente.

La frecuencia máxima f_{Nyq} corresponde bajo (1.13) a una frecuencia máxima en la escala de Mel que denotamos simplemente por

$$\phi_{max} = 2595 \log_{10}\left(1 + \frac{f_{Nyq}}{700Hz}\right).$$

Se divide entonces el rango de frecuencias en la escala de Mel $[0, \phi_{max}]$ en $\mathcal{K}+1$ subintervalos de ancho

$$\Delta_{mel} = \frac{\phi_{max}}{\mathcal{K}+1} \tag{2.6}$$

con las frecuencias centrales

$$\phi_c(k) = (k+1)\Delta_{mel}, \quad k = 0, 1, 2, \ldots, \mathcal{K}-1. \tag{2.7}$$

Las funciones de filtro triangulares en la escala de Mel tendrán como soporte el intervalo de frecuencias $[\phi_c(k) - \Delta_{mel}, \phi_c(k) + \Delta_{mel}] = [\phi_c(k-1), \phi(k+1)]$, con valor máximo de uno en la frecuencia central $\phi_c(k)$, explícitamente

$$T_{k,mel}(\phi) = \begin{cases} 0, & \text{si } |\phi - \phi_c(k)| \geq \Delta_{mel}, \\ 1 + \frac{\phi - \phi_c(k)}{\Delta_{mel}}, & \text{si } \phi_c(k) - \Delta_{mel} \leq \phi \leq \phi_c(k) \\ 1 - \frac{\phi - \phi_c(k)}{\Delta_{mel}}, & \text{si } \phi_c(k) \leq \phi \leq \phi_c(k) + \Delta_{mel} \end{cases} \tag{2.8}$$

En la Figura 2.5 se muestran un banco de filtros de Mel $T_{k,mel}(n)$ para $k = 0, 1$, $\mathcal{K} = 2$ canales. En la Figura 2.6 se muestra el banco de filtros en la escala de Mel (mel) y el banco de filtros en la escala de frecuencia normal (Hz) con $\mathcal{K} = 22$. La Figura 2.7 muestra en detalle la relación entre un filtro uniforme en la escala de Mel y el correspondiente filtro en la escala normal.

Figura 2.5: $\mathcal{K} = 2$ funciones de filtro $T_{k,mel}$, uniformes en la escala de Mel

Figura 2.6: Banco de filtros uniforme en la escala de Mel (izq.) y el mismo banco en la escala normal (Hz)

Ahora será necesario transformar las funciones de filtro uniformes en la escala de Mel (2.8) a la escala lineal (Hz). Partiendo de (2.8), las frecuencias centrales $\phi_c(k)$, $k = 0, 1, 2, \ldots, \mathcal{K} - 1$ se transforman a la escala lineal mediante la transformación inversa de (1.13), es decir

$$f_c(k) = 700 \left(10^{\phi_c(k)/2595} - 1\right)$$

Si las frecuencias lineales están cuantizadas

$$f_n = n\Delta f = \frac{nf_s}{N}$$

entonces las frecuencias centrales en la escala lineal corresponderán a un índice $n_c(k)$ tal que

$$f_c(k) = n_c(k)\Delta f$$

en otras palabras, las frecuencias centrales corresponden al índice de Fourier

$$n_c(k) = \left\lfloor \frac{f_c(k)}{\Delta f} \right\rfloor$$

Los coeficientes de ventana

$$\eta_{kn}, \quad k = 0, 1, 2, \ldots, \mathcal{K} - 1, \quad n = 0, 1, 2, \ldots, N - 1 \tag{2.9}$$

se definen entonces como sigue:

$$\eta_{k,n} = \begin{cases} 0 & \text{si } n \leq n_c(k-1) \text{ o } n \geq n_c(k+1) \\ \frac{n - n_c(k-1)}{n_c(k) - n_c(k-1)}, & \text{si } n_c(k-1) \leq n \leq n_c(k), \\ \frac{n - n_c(k+1)}{n_c(k) - n_c(k+1)}, & \text{si } n_c(k) \leq n \leq n_c(k+1) \end{cases}$$

Figura 2.7: Filtro uniforme en la escala de Mel (vertical) y el correspondiente filtro en la escala normal (horizontal, en Hz)

Figura 2.8: Gráfico rasterizado de la matriz de ventana $\eta_{k,n}$, para $\mathcal{K} = 22$ canales; k es el índice del canal, n el índice de frecuencia lineal.

es decir a lo largo del renglón k, $\eta_{k,n}$ es cero fuera del intervalo $[n_c(k-1), n_c(k+1)]$; se incrementa linealmente de cero en $n_c(k-1)$ hasta un valor máximo de 1 en $n_c(k)$ para después decrecer linealmente a cero en $n_c(k+1)$. La Figura 2.8 muestra un gráfico de la matriz para $N = 256$, $\mathcal{K} = 22$, $f_s = 8000$.

Antes de aplicar la transformación cepstral al espectro de potencia, los coeficientes cepstrales se subdividen en ventanas con la matriz de Mel obteniéndose así .

$$G(k) = \sum_{n=0}^{N-1} \eta_{kn} |V(n)|^2, \qquad k = 0, 1, 2, \ldots, \mathcal{K} - 1. \tag{2.10}$$

Finalmente, el corte para suavizar el espectro se lleva a cabo, seleccionando los primeros $\mathcal{Q}$ coeficientes que constituyen los *coeficientes cepstrales de Mel*

$$c(q) = \sum_{k=0}^{\mathcal{K}-1} \log(G(k)) \cos\left(\frac{\pi q(2k+1)}{2\mathcal{K}}\right), \quad q = 0, 1, 2 \ldots, \mathcal{Q} - 1, \tag{2.11}$$

los cuales son los coeficientes que en resumen, codifican la información de voz para el reconocimiento.

Una propiedad importante de remarcar es que mientras los coeficientes cepstrales $G(k)$ en general están correlacionados estadísticamente, los coeficientes ceptrales $c(q)$ no lo están, lo cual es una propiedad relevante en el reconocimiento de voz. En los gráficos 1.8 se verifica empíricamente esta propiedad.

2.2.1. Coeficientes cepstrales dinámicos

Adicionalmente a los coeficientes cepstrales $c_m(q)$, $q = 0, 1, 2, \ldots \mathcal{Q} - 1$, la transición de los coeficientes de una ventana a otra se cuantifica mediante la diferencia y la segunda diferencia de los coeficientes cepstrales de ventanas consecuitivas.

$$\Delta c_m(q) = c_{m+\tau}(q) - c_{m-\tau}(q) \tag{2.12}$$

$$\Delta^2 c_m(q) = \Delta c_{m+\tau}(q) - \Delta c_{m-\tau}(q) \tag{2.13}$$

los cuales se obtienen de comparar los coeficientes cepstrales moviendo el offset de la ventana τ unidades hacia adelante y hacia atrás. El retraso τ es generalmente pequeño entre 2 y 4, con ajustes convenientes en la ventanas inicial y final.

2.2.2. Energía

La energía de la señal subdividida en ventanas o su logaritmo, se toma como una característica importante de la voz, pues permite distinguir la señal del ruido de fondo. Se define simplemente como la norma cuadrática

$$e = \sum_{k=0}^{N-1} v(k)^2.$$

2.3. Extracción de características

La secuencia de vectores de características de voz se forma con vectores de dimensión 36, consistentes en 12 coeficientes cepstrales $c(q)$, 12 coeficientes de velocidad $\Delta c(q)$, 12 coeficientes de aceleración $\Delta^2 c(q)$ y el logaritmo de la energía $\log(e)$ para cada ventana de 16–25 ms de duración que se recorren cada 10 ms,

$$\begin{aligned}
\widetilde{X} &= \left\{ \vec{X}_0, \vec{X}_1, \ldots, \vec{X}_{T_X-1} \right\}, \\
\vec{X}_m &= (c_m(0), c_m(1), \ldots, c_m(\mathcal{Q}-1), \Delta c_m(0), \Delta c_m(1), \ldots, \Delta c_m(\mathcal{Q}-1), \\
&\qquad \Delta^2 c_m(0), \Delta^2 c_m(1), \ldots, \Delta^2 c_m(\mathcal{Q}-1), \log(e_m)) \,,
\end{aligned}$$

donde T_X es la longitud de la secuencia. Cada instancia de voz del dígito resulta en una secuencia de vectores de características de distinta longitud en general, ya que inclusive un mismo sujeto puede pronunciar la misma voz en distinta duración.

Capítulo 3

Medidas de distancia espectral

Cada vector de características de la voz se representa como un punto en un espacio vectorial d–dimensional. El problema de clasificación de dígitos aislados se puede describir brevemente como sigue: Se coleccionan instancias de cada dígito, por ejemplo grabaciones por un solo sujeto o varios sujetos, los cuales llamaremos vectores prototipo. Las instancias se agrupan en clases que representan a los dígitos. Partiendo de una instancia de voz de un dígito desconocido, hablado por una persona, se quiere determinar a qué clase pertenece. Este es un problema de *clasificación* de patrones. Será necesario primeramente definir una noción de distancia entre vectores. Con más precisión, cada dígito patrón corresponde a una secuencia de vectores de características en nuestro espacio vectorial d-dimensional

$$\widetilde{W} = \left\{ \vec{W}_0, \vec{W}_1, \ldots, \vec{W}_{T_W-1} \right\}, \quad \vec{W}_i \in \mathbb{R}^d$$

de longitud T_W, y el dígito desconocido a la secuencia

$$\widetilde{X} = \left\{ \vec{X}_0, \vec{X}_1, \ldots, \vec{X}_{T_X-1} \right\}, \quad \vec{X}_i \in \mathbb{R}^d.$$

En base a la distancia definida entre vectores de características,

$$d(\vec{X}, \vec{W}),$$

deberá definirse una noción de distancia entre secuencias de vectores de características, en general de distintas longitudes T_W, T_X,

$$D(\widetilde{X}, \widetilde{W}).$$

3.1. Distancia pesada

Dado que el espacio vectorial de características aloja "distintas unidades" es natural proponer la distancia euclideana pesada

$$d(\vec{X}, \vec{W}) = \sum_{i=1}^{d} \lambda_i (X_i - W_i)^2 \tag{3.1}$$

con X_i, W_i, las componentes de los vectores, $\vec{X} = (X_1, X_2, \ldots, X_d)$, $\vec{W} = (W_1, W_2, \ldots, W_d)$. Los pesos λ_i normalizan las unidades y proporcionan una escala natural de dispersión de la i–ésima característica. Para estimar los pesos, partimos de que se dispone de un total de N *muestras o vectores de entrenamiento:*

$$\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{N-1}\}$$

donde $\vec{x}_n = (x_{n,1}, x_{n,2}, \ldots, x_{n,d})$. Se toma entonces la media de los vectores de entrenamiento,

$$\vec{m} = \frac{1}{N} \sum_{n=0}^{N-1} \vec{x}_n$$

y $\lambda_i = 1/\sigma_i^2$, donde σ_i es la varianza estimada, de la i–ésima característica cepstral,

$$\sigma_i = \frac{1}{N-1} \sum_{n=0}^{N-1} (x_{n,i} - m_i), \quad i = 1, 2, \ldots, d.$$

3.2. Distancia de Mahalanobis

Otra distancia de uso común en el reconocimiento de patrones es la distancia de Mahalanobis, la cual toma en cuenta la correlación que existe entre las características cepstrales

$$d_C(\vec{X}, \vec{W}) = (\vec{X} - \vec{W})^T C^{-1} (\vec{X} - \vec{W}), \tag{3.2}$$

donde C^{-1} es la inversa de la matriz de varianzas y covarianzas muestrales

$$C = \frac{1}{N-1} \sum_{n=0}^{N-1} (\vec{x}_n - \vec{m})(\vec{x}_n - \vec{m})^T;$$

explícitamente,

$$C = \begin{bmatrix} \sigma_{1,1} & \cdots & \sigma_{1,j} & \cdots & \sigma_{1,d} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \sigma_{d,1} & \cdots & \sigma_{d,j} & \cdots & \sigma_{d,d} \end{bmatrix},$$

con

$$\sigma_{i,j} = \frac{1}{N-1} \sum_{n=0}^{N-1} (x_{n,i} - m_i)(x_{n,j} - m_j), \quad i, j = 1, 2, \ldots, d.$$

Capítulo 4

Alineación temporal dinámica(DTW)

4.1. Definición de la Tarea de Clasificación

El problema de reconocimiento de dígitos aislados se puede plantear como sigue: Los dígitos se representan por una colección de clases abstractas

$$\Omega = \{\omega_0, \omega_2, \ldots, \omega_{V-1}\}.$$

Cada clase ω_v representa una collección de K_{ω_v} secuencias de vectores prototipo

$$\widetilde{W}_{k,\omega_v}, \quad k = 0, 1, \ldots, K_{\omega_v} - 1.$$

Cada $\widetilde{W}_{k,\omega_v}$ es una secuencia de vectores de características cepstrales de duración $T_{W_{k,\omega_v}}$

$$\widetilde{W}_{k,\omega_v} = \{\vec{W}_0, \vec{W}_1, \ldots, \vec{W}_{T_{W_{k,\omega_v}}-1}\}.$$

Para simplificar la notación, denotaremos una secuencia de vectores prototipo por

$$\widetilde{W} = \{\vec{W}_0, \vec{W}_1, \ldots, \vec{W}_{T_W-1}\}.$$

y para especificar su clase diremos que $\widetilde{W} \in \omega_v$. Sea

$$\widetilde{X} = \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_n\}$$

la secuencia de vectores de un dígito desconocido. Si $D(\widetilde{W}, \widetilde{X})$ es una distancia entre secuencias de vectores, el problema de clasificación se plantea como

$$\text{mín}\, D(\widetilde{X}, \widetilde{W}_{k,\omega_v}) = D(\widetilde{X}, \widetilde{W}_{k^*,\omega_v^*}) \Longleftrightarrow \widetilde{X} \in \omega_v^*. \tag{4.1}$$

Figura 4.1: Alineamiento temporal

4.2. El Problema de Alineamiento Temporal

Una distancia entre secuencias de vectores deberá cubrir algunos requisitos:

1. Deberá medir la distancia entre dos secuencias de vectores de distinta longitud temporal T_X y T_W.

2. Deberá hacer una asignación óptima en algún sentido entre vectores individuales de las secuencias. Esto significa alinear temporalmente las secuencias.

3. La distancia total deberá ser la suma de las distancias entre vectores individuales óptimamente asignados.

El problema principal para definir la distancia entre secuencias de vectores es hallar el asignamiento óptimo entre los vectores individuales. En la Figura 4.1 podemos ver dos secuencias de vectores $\widetilde{X}$ y $\widetilde{W}$ de longitudes 6 y 8 respectivamente. La secuencia $\widetilde{W}$ está rotada 90°, así el índice temporal para esta sucesión corre verticalmente de abajo a arriba. La duración de las secuencias define la retícula formada por el producto cartesiano[1] $\bar{6} \times \bar{8}$. Cada trayectoria a través de esta retícula, tal como la mostrada en la Figura 5.1, representa una posible asignación de las parejas de vectores. Por ejemplo para la trayectoria mostrada en la Figura

$$g = \{(0,0),(1,1),(2,2),(3,2),(4,2),(5.3),(6,4),(7,4)\} \equiv \{g_1, g_2, \ldots, g_{L_g}\}$$

significa que se hace la asignación

$$\begin{aligned}
\vec{X}_0 &\leftrightarrow \vec{W}_0 \\
\vec{X}_1 &\leftrightarrow \vec{W}_1 \\
\vec{X}_2 &\leftrightarrow \vec{W}_2 \\
\vec{X}_3 &\leftrightarrow \vec{W}_2 \\
\vec{X}_4 &\leftrightarrow \vec{W}_2 \\
\vec{X}_5 &\leftrightarrow \vec{W}_3 \\
\vec{X}_6 &\leftrightarrow \vec{W}_4 \\
\vec{X}_7 &\leftrightarrow \vec{W}_4
\end{aligned}$$

[1]Para un natural n, abreviamos $\bar{n} = \{0, 1, \ldots, n-1\}$.

Observe que las trayectorias admisibles deben ser monótonas ya que en cualquier paso de la alineación temporal el tiempo no retrocede. El segmento de trayectoria constante en el ejemplo, vgr., $(2,2),(3,2),(4,2)$ significa que el tiempo se detiene en 2 para la secuencia $\widetilde{W}$, mientras de se deja avanzar en la secuencia $\widetilde{X}$.

Cada trayectoria tiene asociada una distancia acumulada

$$D(\vec{X},\vec{W},g)=\sum_{\ell=1}^{L_g} d(\vec{X}_{g_\ell(1)},\vec{W}_{g_\ell(2)})$$

y definimos la distancia entre las secuencias como

$$D(\widetilde{X},\widetilde{W})=\min_g D(\vec{X},\vec{W},g). \tag{4.2}$$

4.3. El principio de Bellman

Determinar el mínimo de las distancias acumuladas de acuerdo a (4.2) es un problema computacionalmente complejo, ya que el número de trayectorias posibles en un reticulado de $\bar{n}\times\bar{m}$ es del orden de $(m-1)\times 3\times 3\times\cdots\times 3$ ($n-1$ factores iguales a 3) $=(m-1)3^{n-1}$, ya que el primer nodo g_1 de la trayectoria se puede elegir de m maneras; como la trayectoria es monótona y continua, el segundo nodo g_2 se puede elegir de 3 maneras, si $g_1(2)<m$; si $g_1(2)=m$ el siguiente nodo se puede elegir de una sola manera.

Debido a la complejidad computacional en buscar sobre todas las posibles trayectorias del reticulado $\bar{n}\times\bar{m}$, el principio de Bellman consiste en optimizar la trayectoria local antecesora de cada punto de la retícula. Para ello consideremos un nodo (i,j) como se muestra en la Figura 4.2; debido a la monotonía, los nodos antecesores sólo pueden ser: $(i-1,j)$, $(i-1,j-1)$ o $(i,j-1)$. La función predecesora $\psi(i,j)$ asigna el nodo antecesor que tenga peso acumulado mínimo; es decir si $\psi(i,j)=(i',j')$ y denotamos por $\delta(i-1,j),\delta(i,j-1),\delta(i-1,j-1)$ los pesos acumulados de las trayectorias óptimas que llegan a cualquiera de los posibles nodos antecesores entonces

$$\delta(i,j) = \min\begin{cases}\delta(i,j-1)+d(\vec{W}_i,\vec{X}_j)\\ \delta(i-1,j)+d(\vec{W}_i,\vec{X}_j)\\ \delta(i-1,j-1)+d(\vec{W}_i,\vec{X}_j)\end{cases} \tag{4.3}$$

$$\psi(i,j) = \arg\min\begin{cases}\delta(i,j-1)+d(\vec{W}_i,\vec{X}_j)\\ \delta(i-1,j)+d(\vec{W}_i,\vec{X}_j)\\ \delta(i-1,j-1)+d(\vec{W}_i,\vec{X}_j)\end{cases} \tag{4.4}$$

Figura 4.2: Principio de Bellman

Figura 4.3: (a) Trayectorias extremas. (b) Espacio de trayectorias admisibles restringido.

Es importante resaltar que la recursión definida por (4.3-4.4) está definida a través de las trayectorias *óptimas* que llegan a los posibles nodos antecesores. El problema de optimización está completamente determinado si se especifica

- La clase de trayectorias admisibles.
- Las condiciones de frontera de la trayectoria óptima.
- La función de costo.

Respecto de las trayectorias admisibles, intuitivamente es claro que las trayectorias extremas como las mostradas en la Figura 4.3 son poco probables, de ahí que la búsqueda puede hacerse más eficiente restringiendo las trayectorias a estar sobre una región sombreada. Las condiciones de frontera implican cierto conocimiento del alineamiento en los extremos. Por ejemplo si las trayectorias tienen extremo inicial fijo, por ejemplo $g_1 = (0, 0)$ para todas las trayectorias admisibles, significa que estamos seguros que los inicios de voz de las secuencia $\widetilde{X}$ y $\widetilde{W}$ están perfectamente definidos y por lo tanto se alinean temporalmente. La elección del extremo final de las trayectorias admisibles, $g_L = (T_X, T_W)$ se interpreta de manera análoga. Si por ejemplo, no estuviéramos seguros de la ventana en la que termina la voz desconocida $\widetilde{X}$ se podría dejar el extremo final libre en la clase de trayectorias admisibles.

La detección del inicio y término de voz se conoce como el problema de detección de extremos de voz, es un problema complejo, ya que involucra distinguir el ruido de fondo de los posibles chasquidos que pueden formar parte de la voz. En este trabajo esto se ha resuelto con una edición cuidadosa de las grabaciones antes de entrar al módulo de preprocesamiento. Rabinier y Sambur [12] proponen un algoritmo que evalua el nivel de energía y la razón de cruces por cero de la señal para la detección de extremos. En resumen, en este trabajo se supone que cualquier trayectoria admisible satisface las condiciones de frontera:

$$g_1 = (0, 0), \qquad g_L = (T_X - 1, T_W - 1). \tag{4.5}$$

La función de costo se toma como la suma de las distancias acumuladas entre los vectores individuales. Para una trayectoria parcial $\{g_1, g_2, \ldots, g_m\}$ se le asignará el costo

$$\sum_{\ell=1}^{m} d(\vec{X}_{g_1(\ell)}, \vec{W}_{g_1(\ell)}). \tag{4.6}$$

Así una trayectoria de gran peso presenta más disimilitudes cepstrales en las ventanas alineadas temporalmente, o de manera alternativa, una trayectoria de peso mínimo alinea temporalmente las ventanas de modo que su disimilitud cepstral sea mínima.

Remarcamos que en (4.3) los valores δ representan los pesos acumulados de las trayectorias *óptimas* parciales que llegan a los posibles nodos antecesores y la función de elección ψ selecciona simplemente el antecesor y por lo tanto continua la trayectoria local óptima que llega al nodo (i, j) y es por tanto un algoritmo recursivo. Para que la recurrencia (4.3-4.4) esté bien definida habrá que definir $\delta(0, 0)$ y $\psi(0, 0)$. Una elección común es poner

$$\delta(0, 0) = -\infty, \quad \psi(0, 0) = -1 \tag{4.7}$$

El algoritmo (4.3-4.4) junto con los valores en el paso cero (4.7) permiten resolver el problema de la búsqueda de la trayectoria óptima por *retropropagación* como sigue: partir de $g_L^* = (T_X - 1, T_W - 1)$ se puede seguir hacia atras la trayectoria óptima $g_{L-1}^* = \psi(g_L^*)$, $g_{L-2}^* = \psi(g_{L-1}^*)$, y en general $g_{\ell-1}^* = \psi(g_\ell)$, mientras $\psi(g_\ell) \neq (-1, 1)$.

Capítulo 5

Implementación DTW

En este capítulo se describen los dos principales módulos del sistema de reconocimiento de dígitos aislados basados en DTW: el módulo de preprocesamiento y el módulo de reconocimiento. Se comenzó con una implementación en Mathematica para posteriormente migrar a Matlab y hacer el código más eficiente. Se aprovecho principalmente la posibilidad de guardar los archivos binarios *.mat conteniendo las secuencias de vectores de propiedades cepstrales en Matlab.

5.1. Módulo de preprocesamiento

La Figura 5.1 ilustra la estructura del módulo de preprocesamiento. Consta de varios submódulos que ahí se indican. De acuerdo a (2.1) la señal se enfatiza previo al cálculo del espectro de potencia para eliminar el decaimiento de -6 dB por octava. La subdivisión en ventanas se lleva a cabo usando la función de Hamming (2.2). El espectro de potencia se calcula tomando la transformada discreta de Fourier y después tomando el módulo al cuadrado. Tomando 10 veces el logaritmo se obtiene el espectro en unidades de decibelios. Un banco de filtros de Mel se usa para tomar en cuenta la respuesta no lineal del oído humano, lo cual se resume en el cálculo de la matriz de coeficientes de Mel (2.8). Finalmente, en vez de tomar la transformada inversa de Fourier del espectro de potencia, se usa la transformada discreta de Fourier en cosenos que aprovecha la simetría del espectro de potencias (2.5).

A continuación se describen la funciones principales del módulo de preprocesamiento en Matlab.

SequenceFeatureVector

Figura 5.1: Módulo de preprocesamiento. DTFT: transformada de Fourier en tiempo discreto. CDFT: Transformada en coseno de tiempo discreto. MFCC: coeficientes cepstrales en la escala de MEl.

Función: Calcula la secuencia de vectores de atributos para un archivo *.wav

Sintaxis
`X=SequenceFeatureVector(soundfile,lengthWindowMs,shiftWindowMs,...`

Parámetros:
`soundfile`: archivo de sonido leido previamente con wavread.
`lengthWindowMs`: Tamaño de la ventana en ms.
`shiftWindowMs`: corrimiento de la ventana en ms.
`sampleRate:` Frecuencia de muestreo en Hz.
`noMelCoefs`: número de coeficientes de Mel.
`noChannels`: número de canales.

5.1.1. Funciones que son llamadas por **SequenceFeatureVector**

ParametersOfSignal

Función: Calcula parámetros de la señal de sonido

Sintaxis:
```
[L,wlength,wshift,nwin]=ParametersOfSignal(soundfile, lengthWindowMs,...
                            shiftMs, samplefrequency)
```

Parámetros:
`soundfile`: archivo de sonido leido previamente con wavread.
`lengthWindowMs`: tamaño de la ventana en ms (16–25 ms).
`shiftMs`: cambio de la ventana (10–15 ms).
`samplefrequency`: frecuencia de muestreo.
`L`: longitud de la señal.
`wlength`: longitud de la ventana en unidades de período de muestreo.
`wshift`: corrimiento en unidades enteras.
`nwin`: número de ventanas que caben en el segmento.

Enfasis

Función: Elimina el decaimiento de -6dB por octava

Sintaxis:
`senf=Enfasis(soundfile)`

Parámetros:
`soundfile`: archivo de sonido, ya como lista de números.
`senf`: archivo de sonido enfatizado

SFT

Función: Calcula el espectro de potencia.

Sintaxis:
`V=SFT(signal,offset,wlength)`

Parámetros:
`signal`: vector que contiene una señal enfatizada.
`offset`: entero que indica donde comienza la ventana.
`wlength`: longitud de la ventana.
`V`: espectro de potencia o cuadrado de la transformadad de Fourier de signal.

MFCC

Función: Calcula los coeficientes cepstrales de la frecuencia de Mel.

Sintaxis
`c=MFCC(noMelCoefs,noChannels,lengthWindow,powerSpectrum,sampleRate)`

Parámetros:
`noMelCoefs`: número de coeficientes de Mel.
`noChannels`: número de canales.
`lengthWindow`: longitud de la ventana (entero).
`powerSpectrum`: espectro de potencia.
`sampleRate`: frecuencia de muestreo.
`c`: coeficientes cepstrales de la frecuencia de Mel.

melMatrix

Función: Calcula la matriz de los coeficientes de Mel

Sintaxis
`n=melMatrix(lengthWindow,noChannels,sampleRate)`

Parámetros:

`lengthWindow`: longitud de la ventana (entero).
`noChannels`: número de canales.
sampleRate: frecuencia de muestreo.
`n`: matriz de los coeficientes de Mel.

Energy

Función: Calcula la energía de la señal.

Sintaxis
`e=Energy(signal)`

Parámetros:
`signal`: es la señal
`e`: energía de la señal.

Dc

Función: Calcula la derivada de los coeficientes cepstrales de la frecuencia de Mel.

Sintaxis
`dc=Dc(signal, offset, tauplus, tauminus, noMelCoefs, noChannels,... lengthWindow, samp`

Parámetros:
`signal:` la señal.
`offset`: es el inicio de ventana (un entero)
`tauplus`: índice de tiempo en el futuro, usualmente un entero entre 2 y 4.
`tauminus`: índice de tiempo en el pasado, usualmente un entero entre 2 y 4.
`noMelCoefs`: número de coeficientes cepstrales de Mel.
`noChannels`: número de canales.
`lengthWindow`: longitud de la ventana (calculada por ParametersOfSignal).
`sampleRate`: frecuencia de muestreo.
`dc`: derivada de los coeficientes cepstrales de Mel.

DDc

Función: Calcula la segunda derivada de los coeficientes cepstrales de Mel.

Sintaxis
`ddc=DDc(signal, offset, tauplus, tauminus, noMelCoefs, noChannels,.. lengthWindow, sam`

Parámetros:
`signal`: es la señal (como vector).

Figura 5.2: Módulo de clasificación

`offset`: es el inicio de ventana (un entero).
`tauplus`: índice de tiempo en el futuro, usualmente un entero entre 2 y 4.
`tauminus`: índice de tiempo en el pasado, usualmente un entero entre 2 y 4.
`noMelCoefs`: número de coeficientes cepstrales de Mel.
`noChannels`: número de canales.
`lengthWindow`: longitud de la ventana (calculada por ParametersOfSignal).
`sampleRate`: frecuencia de muestreo.
`ddc`: la segunda derivada de los coeficientes cepstrales de Mel.

5.2. Módulo de Clasificación basado en DTW

La Figura 5.2 muestra la estructura del módulo de clasificación. Las secuencias patrón fueron generadas con el módulo de preprocesamiento y guardadas en archivos *.mat que son leídos en el módulo de clasificación. El bloque DTW hace el trabajo de alineamiento temporal y cálculo de la trayectoria óptima. El reconocimiento es simplemente la identificación a través de algún código, como el nombre del archivo, de la secuencia patrón a una distancia mínima de la secuencia del dígito desconocido.

La mayor parte del reconocmiento se hace en un programa principal que llama a funciones auxiliares que calculan distancias y que se describen a continuación.

Distance

Función: Hace la comparación entre 2 secuencias de vectores asociados a 2 señales, mediante alineamiento temporal DTW.

Sintaxis
`Dst=Distance(W, X)`

Parámetros: `W, X`: secuencia de vectores de características en forma de matrices, las ventanas se identifican por renglones, las características cepstrales por columnas.
`Dst`: distancia acumulada óptima.

d

Función: Calcula la distancia euclidiana, pesada o de Mahalanobis entre esos 2 vectores de características.

Sintaxis
`dis=d(Wv,Xv)`

Parámetros:
`Wv`: vector dígitos-patrón
`Xv`: vector dígitos-prueba
`dis`: distancia.

Capítulo 6

Modelos de Markov Ocultos

6.1. Definiciones básicas

Los HMM son un modelo de sistema dinámico estocástico parcialmente observado (que se ve en teoría de sistemas y en teoría de control). El término HMM es frecuentemente restringido a modelos con estados y medidas en un conjunto discreto y en tiempo discreto. Los HMM se usan en los sistemas dinámicos estocásticos, en el procesamiento de señales y en sistemas de control entre otros; Las formas avanzadas de procesamiento de señales, que tienen aplicación en ingeniería y ciencias, particularmente en situaciones cuando los modelos de señales son únicamente parcialmente conocidos y en medio ambiente ruidosos. Los siguientes autores introdujeron los HMM : Lakhdar Aggoun con procesamiento de señales (1993), Vikram Krishnamurthy y Lige Xia con el conjunto de pasos para la adaptación del procesamiento de señales de los HMM , y Hailiang Yang con filtros de dominio de tiempo continuo, con su observación a tiempo discreto.

Definición 1. *Sea $(\Omega, \mathcal{F}, P)$ un espacio de probabilidad y S un conjunto no vacío, finito. Una sucesión de variables aleatorias*

$$\{q_n \colon \Omega \to S, \quad n = 1, 2, ...\}$$

se llama cadena de Markov con espacio de estados S si satisface la condición de Markov, esto es, si $\forall n \geq 1$ y toda sucesión $S_1, S_2, ..., S_{n-2} \in S$ se cumple

$$P(q_n = S_n \mid q_{n-1} = S_n - 1, ..., q_1 = S_1) = P(q_n = S_n \mid q_{n-1} = S_{n-1}).$$

La distribución de q_1 se llama distribución inicial y la denotaremos por π.

Informalmente, la cadena de Markov oculta consta de 2 procesos: una cadena de Markov, que generalmente no podemos observar, y un proceso observable, con ciertas características, mediante el cual se espera obtener información acerca de la cadena de Markov, de ahí el nombre del proceso. Por simplicidad, se puede considerar un espacio de estados finito tanto para el proceso oculto como para el observable.
Un HMM esta caracterizado por lo siguiente:

1. N, el número de estados en el modelo. Aunque los estados son ocultos, para muchas aplicaciones prácticas hay a menudo algunas de significancia física unidas a los estados o al conjunto de estados en el modelo. Generalmente los estados son interconectados en un camino a cualquier estado que puede ser alcanzado por cualquier otro estado; sin embargo veremos mas adelante en este trabajo que hay otras posibles interconexiones que a menudo son de interés. Denotamos los estados individuales como $S = \{S_1, ..., S_N\}$, y los estados al tiempo t como q_t.

2. M, el número de distintas observaciones simbolizado por el estado, i.e., el tamaño del alfabeto discreto. Los simbolos de observación corresponden a la salida física del sistema a ser modelado.

3. La distribución de probabilidad de transición de los estados $A = \{a_{ij}\}$ donde $a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i), \quad 1 \leq i, j \leq N$. Para el caso especial donde cualquier estado puede alcanzar cualquier otro estado en un paso singular, tenemos $a_{ij} > 0 \ \forall i, j$. Para otros tipos de modelos de Markov, podemos tener $a_{ij} = 0$ para uno ó mas pares de (i,j)

4. El símbolo de observación de la distribución de probabilidad en el estado j, $B = \{b_j(k)\}$, donde

   $b_j(k) = P(v_k \text{ al tiempo } t \mid q_t = S_j), \quad 1 \leq j \leq N \quad 1 \leq k \leq M$

5. La distribución del estado inicial $\pi = \{\pi_i\}$ donde $\pi_i = P(q_1 = S_i)$, $1 \leq i \leq N$.

   Dando los valores apropiados de $N, M, A, B, \text{y } \pi$, los HMM pueden ser usados como un generador para dar una sucesión de observación

   $$O = O_1 O_2 ... O_t$$

   (donde cada observación O_t es uno de los simbolos para V_t y T es el número de observaciones en la sucesión) como sigue:

a) Elegimos un estado inicial $q_1 = S_i$ de acuerdo a la distribución de estado inicial π.

b) Sea $t = 1$.

c) Elegimos $O_t = v_k$ de acuerdo al símbolo de la distribución de probabilidad en el estado S_i, i.e., $b_i(k)$.

d) Transitamos a un nuevo estado $q_{t+1} = S_j$ de acuerdo a la transición de estado de la distribución de probabilidad para el estado S_i, i.e., a_{ij}.

e) Sea $t = t + 1$; regresamos al paso c) si $t < T$ en otro caso termina el procedimiento.

Para determinar un HMM se requiere la especificación de dos parámetros del modelo (N y M), la especificación de los símbolos de observación y la especificación de las tres medidas de probabilidad $A, B, \text{y}\, \pi$. Usaremos la notación $\lambda = (A, B, \pi)$ indica al conjunto de parámetros completo del modelo.

6.2. Los tres problemas básicos de los HMM

1. Dada una sucesión de observaciones $O = O_1O_2...O_T$, y un modelo $\lambda = (A, B, \pi)$ ¿cómo calculamos eficientemente la $P(O|\lambda)$?

2. Dada la sucesión de observaciones $O = O_1O_2...O_T$, y el modelo λ, ¿cómo debemos elegir una sucesión de estados correspondiente $Q = q_1q_2...q_T$ que sea óptima en algún sentido significativo (la que mejor "explique" las observaciones)?

3. ¿Cómo debemos ajustar los parámetros del modelo $\lambda = (A, B, \pi)$ para maximizar la $P(O \mid \lambda)$?

El problema 1 es la evaluación del problema, i.e., que dado un modelo y una sucesión de observaciones, como calculamos la probabilidad de que la sucesión observada es producida por el modelo?
El problema 2 es aquel en el cual intentamos descubrir la parte oculta del modelo, i.e., hallar la sucesión de estados "correcta". Entonces para situaciones prácticas, usualmente empleamos un criterio de optimización que resuelve este problema como el más posible. Típicamente podemos aprender sobre la estructura del modelo, para hallar la sucesión de estados óptima para el reconocimiento continuo del habla, ó para dar promedios estadísticos de estados individuales.

El problema 3 es aquel en el cual intentamos optimizar los parámetros del modelo, así como la mejor descripción, como una sucesión de observación dada. La sucesión de observación usada para ajustar los parámetros del modelo es llamada una sucesión entrenada y se dice que se "entrena" el HMM.

6.3. Soluciones de los tres problemas básicos de los HMM

Solución al problema 1. Deseamos calcular la probabilidad de la sucesión de observación,

$$O = O_1 O_2 ... O_T$$

dado el modelo λ, i.e., $P(O \mid \lambda)$. Debemos enumerar cada sucesión de estados posibles de longitud T (el número de observaciones). Consideremos una tal sucesión de estados fija

$$Q = q_1 q_2 ... q_T \tag{6.1}$$

donde q_1 es el estado inicial. La probabilidad de la sucesión de observación O para la sucesión de estados (6.1) es

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

donde estamos asumiendo una independencia estadística de las observaciones. Obtenemos

$$P(O \mid Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdot \ldots \cdot b_{q_T}(O_T).$$

La probabilidad de tal sucesión de estado Q puede ser escrita como

$$P(Q \mid \lambda) = \pi_{q_1} \mathbf{a}_{q_1 q_2} \mathbf{a}_{q_2 q_3} \cdots \mathbf{a}_{q_{T-1} q_T}$$

La probabilidad de que O y Q ocurran simultáneamente, es simplemente el producto de los 2 términos de arriba, i.e.,

$$P(O, Q \mid \lambda) = P(O \mid Q, \lambda) P(Q \mid \lambda)$$

La probabilidad de O (dado el modelo) se obtiene por sumas de estas probabilidades simultáneas sobre todas las posibles sucesiones de estado q

$$P(O \mid \lambda) = \sum_{\text{todos los } Q} P(O \mid Q, \lambda) P(Q \mid \lambda) \tag{6.2}$$

$$= \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(O_1) \mathrm{a}_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T). \tag{6.3}$$

La interpretación de los cálculos en la ecuación de arriba es la siguiente: Inicialmente (al tiempo $t = 1$) estamos en el estado q_1 con probabilidad π_{q_1}, y generamos el símbolo O_1 (en este estado) con probabilidad $b_{q_1}(O_1)$. El tiempo aumenta de t a $t + 1(t = 2)$ y hacemos una transición al estado q_2 desde el estado q_1 con probabilidad $a_{q_1 q_2}$, y generamos el símbolo O_2 con probabilidad $b_{q_2}(O_2)$. El proceso continua de esta manera hasta que transitamos (al tiempo T) del estado q_{T-1} al estado q_T con probabilidad $a_{q_{T-1} q_T}$ y generamos el símbolo O_T con probabilidad $b_{q_T}(O_T)$.
El cálculo de $P(O|\lambda)$, de acuerdo a la definición directa (6.3), requiere del orden de $2TN^T$ cálculos; entonces, para cada $t = 1, 2, ..., T$, hay N estados posibles que pueden ser alcanzados (i.e., hay N^T sucesiones de estados posibles). Necesitamos $(2T-1)N^T$ multiplicaciones y $N^T - 1$ adiciones.
Estos cálculos son computacionalmente no factibles. Claramente se necesita un procedimiento más eficiente para resolver el problema 1, y este es el procedimiento forward-backward (bfp).

El procedimiento forward–backward
Estrictamente hablando, únicamente necesitamos la parte del forward del bfp para resolver el problema 1. Introduciremos la parte backward del bfp en este inciso y después lo usaremos para que ayude a resolver el problema 3.
Consideremos la variable forward $\alpha_t(i)$ definida como

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i \mid \lambda)$$

i.e., la probabilidad de la sucesión de observación parcial, $O_1 O_2 \cdots O_t$, (hasta el tiempo t) y el estado S_i al tiempo t, dado el modelo λ. Podemos resolver inductivamente para $\alpha_t(i)$ como sigue:

1. Inicialización:
$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

2. Inducción:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1 \quad 1 \leq j \leq N \tag{6.4}$$

3. Terminación:
$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

Análogamente, podemos considerar una variable backward $\beta_t(i)$ definida como

$$\beta_t(i) = P(O_{t+1}O_{t+2}\cdots O_T \mid q_t = S_i, \lambda).$$

De nuevo se puede resolver $\beta_t(i)$ inductivamente:

1. Inicialización:
$$\beta_t(i) = 1, \quad 1 \leq i \leq N$$

2. Inducción:
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1})\beta_{t+1}(j), \quad t = T-1, T-2, ..., 1, \quad 1 \leq i \leq N \tag{6.5}$$

Observación. El procedimiento backward se usará en la solución del problema 3, y este no es requerido para la solución del problema 1.

Solución al problema 2. En el problema 1 se puede dar una solución exacta, en cambio en el problema 2, hay varios posibles caminos, a saber hallando la sucesión de estados "óptima" asociada con la sucesión de observación dada. La dificultad yace en la definición de sucesión óptima de estados; i.e., hay varios criterios posibles de optimalidad. Por ejemplo, un criterio posible de optimalidad es elegir los estados q_t que son individualmente más probables. Estos criterios de optimalidad maximizan el número esperado de los estados individuales correctos. Para implementar esta solución al problema 2, definimos la variable

$$\gamma_t(i) = P(q_t = S_i \mid O, \lambda) \tag{6.6}$$

i.e., la probabilidad de empezar en el estado S_i al tiempo t, dada la sucesión de observación O, y el modelo λ. La ecuación (6.5) puede ser expresada simplemente en términos de las variables forward-backward, i.e.,

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{6.7}$$

Algoritmo de Viterbi

En la mayoría de los casos es suficiente con encontrar solo la mejor secuencia y su probabilidad asociada. Con este fin existen algoritmos que permiten ahorrar muchos cálculos y entre ellos, uno de los más utilizados es el algoritmo de Viterbi. En este

algoritmo la idea central es recorrer el diagrama de transiciones de estados a traves del tiempo, almacenando para cada estado solamente la máxima probabilidad acumulada y el estado anterior desde el que se llega con esta probabilidad.
La máxima probabilidad acumulada se obtiene multiplicando la probabilidad de observación del estado por la máxima probabilidad acumulada entre todos los caminos que llegan hasta él. Para hallar la mejor sucesión de estados individual, $Q = q_1 q_2 ... q_T$ necesitamos definir la cantidad

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

i.e., $\delta_t(i)$ es el mejor resultado (de mayor probabilidad) para un modelo individual al tiempo t, que explica para las primeras t observaciones y finaliza en el estado S_i.

Solución al problema 3
El tercero, y por mucho el problema más difícil de los HMM es el de determinar un método que ajuste los parámetros del modelo (A, B, π) que maximice la probabilidad de observación de una secuencia de observaciones dado el modelo. Para ello no se conoce una solución analítica del modelo que maximice la probabilidad de la secuencia de observaciones.
Podemos sin embargo elegir $\lambda = (A, B, \pi)$ tal que $P(O|\lambda)$ sea localmente maximizada usando un procedimiento iterativo tal como el método de Baum-Welch (ó equivalentemente el método de expectación-maximización (EM)) , ó usando técnicas de descenso por gradiente.
Para el método de Baum primero definimos $\xi_t(i, j)$, la probabilidad de empezar en el estado S_i al tiempo t, y el estado S_j al tiempo $t + 1$, dado el modelo y la sucesión de observación, i.e.,

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda).$$

Debe ser claro de las definiciones de las variables forward y backward, que podemos escribir $\xi_t(i, j)$ en la forma

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)} \tag{6.8}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}, \tag{6.9}$$

donde el número de términos es $P(q_t = S_i, q_{t+1} = S_j, O \mid \lambda)$ y la división por $P(O \mid \lambda)$ nos da la medida de probabilidad deseada.

Se había definido $\gamma_t(i)$ como la probabilidad de empezar en el estado S_i al tiempo t, dada la sucesión de observación y el modelo; hay una relación entre $\gamma_t(i)$ y $\xi_t(i,j)$ dada por

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

Se tienen las siguientes interpretaciones:

$\sum_{t=1}^{T-1} \gamma_t(i)$ = número esperado de tiempos que el estado S_i es visitado.

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = número esperado de transiciones del estado S_i al estado S_j

Usando las formulas anteriores, se puede dar un método de *reestimación de los parámetros de un HMM.* Un conjunto de formulas de reestimación razonable para π, A, y B son:

$$\bar{\pi}_i = \text{frecuencia esperada en el estado } S_i \text{ al tiempo}(t=1) = \gamma_1(i) \tag{6.10}$$

$$\begin{aligned} a_{ij} &= \frac{\text{No. esperado de transiciones del estado } S_i \text{ al estado } S_j}{\text{No. esperado de transiciones del estado } S_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \tag{6.11}$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\text{No. esperado de tiempos en el estado } j \text{ y observando el símbolo } v_k}{\text{No. esperado de tiempos en el estado } j} \\ &= \frac{\sum_{t=1}^{T} \gamma_t(j) \quad \text{t.q. } O_t = v_k}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned} \tag{6.12}$$

Si definimos el modelo actual como $\lambda = (A, B, \pi)$ y consideramos las ecuaciones (6.10)-(6.12) como reestimaciones $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, fué probado por Baum y sus colegas que: (1) el modelo inicial λ define un punto crítico de la función de verosimilitud, en cuyo caso $\lambda = \bar{\lambda}$; ó (2) el modelo $\bar{\lambda}$ es más probable que el modelo λ en el sentido de que $P(O \mid \bar{\lambda}) > P(O \mid \lambda)$, i.e.,que se ha encontrado un nuevo modelo $\bar{\lambda}$ de la cual la sucesión de observación es más probable que tenga que ser producida.
Si iteramos usando $\bar{\lambda}$ en lugar de λ y repetimos el cálculo de reestimación, entonces podemos mejorar la probabilidad de O observado del modelo, hasta que sobrepasemos

una verosimilitud deseada. El resultado final de este procedimiento de reestimación es llamado una *estimación de máxima verosimilitud de los HMM* y será discutida con más detalle más adelante.
Las fórmulas de reestimación (6.10)-(6.12) pueden ser derivadas directamente maximizando La función auxiliar de Baum:

$$Q(\lambda, \bar{\lambda}) = \sum_{Q} P(Q \mid O, \lambda) \log[P(O, Q \mid \bar{\lambda})] \tag{6.13}$$

sobre $\bar{\lambda}$. De nuevo fue probado por Baum y sus colegas que la maximización de $Q(\lambda, \bar{\lambda})$ conduce a una probabilidad incrementada, i.e.

$$\max_{\bar{\lambda}}[Q(\lambda, \bar{\lambda})] \Rightarrow P(O \mid \bar{\lambda}) \geq P(O \mid \lambda).$$

Eventualmente la función de probabilidad converge hacia un punto crítico.

6.3.1. Método de Experanza–Maximización (EM)

Sea $p(x|\Theta)$ una función de densidad gobernada por el conjunto de parámetros Θ. Consideremos un conjunto de datos $\mathcal{X}$ de tamaño N, $\mathcal{X} = \{x_1, \ldots, x_N\}$. Asumimos que estos datos son vectores independientes e identicamente distribuidos (iid) con distribución p. Por lo tanto, la densidad resultante para la muestra es

$$p(\mathcal{X} \mid \Theta) = \prod_{i=1}^{N} p(x_i \mid \Theta) = \mathfrak{L}(\Theta \mid \mathcal{X}).$$

Esta función $\mathfrak{L}(\Theta \mid \mathcal{X})$ es llamada la verosimilitud de los parámetros dados los datos (función de verosimilitud). La verosimilitud es una función de los parámetros Θ donde los datos $\mathcal{X}$ son fijos. En el problema clásico de máxima verosimilitud, la meta es hallar el valor de Θ que maximice $\mathfrak{L}$. Es decir queremos hallar un Θ^* tal que

$$\Theta^* = \arg\max_{\Theta} \mathfrak{L}(\Theta \mid \mathcal{X}).$$

Frecuentemente maximizamos $\log(\mathfrak{L}(\Theta \mid \mathcal{X}))$ porque es más fácil de analizar.
El algoritmo EM es un método general para hallar la máxima verosimilitud de los parámetros de una distribución de un conjunto de datos, cuando estos son incompletos ó tiene valores faltantes.[3]
Hay dos aplicaciones importantes de el algoritmo EM. Una es cuando los datos son verdaderamente faltantes, debido a problemas o limitaciones del proceso de observación.

Otra es cuando la optimización de la función de verosimilitud es analíticamente intratable, sin embargo tal función puede ser simplificada suponiendo la existencia de valores adicionales aún con parámetros faltantes u ocultos. Esta última aplicación es más común en problemas de reconocimiento de voz.
Suponemos que los datos $\mathcal{X}$ son observados y generados por una misma distribución. LLamamos a $\mathcal{X}$ los datos incompletos. Suponemos que existe un conjunto de datos completos $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ y también suponemos una función de densidad conjunta:

$$p(z|\Theta) = p(x, y|\Theta) = p(y|x, \Theta)p(x|\Theta).$$

La función de densidad marginal es $p(x|\Theta)$, estamos suponiendo que hay variables ocultas y conjeturamos los valores de los parámetros, debemos suponer una relación conjunta entre los valores observados y los faltantes.
Con esta nueva función de densidad, podemos definir una nueva función de verosimilitud, $\mathfrak{L}(\Theta \mid \mathcal{Z}) = \mathfrak{L}(\Theta \mid \mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y} \mid \Theta)$, llamada la verosimilitud de los datos–completos. La verosimilitud original $\mathfrak{L}(\Theta \mid \mathcal{X})$ se llama la función de verosimilitud de los datos-incompletos.
El algoritmo EM primero halla el valor esperado de la función log-verosimilitud de los datos-completos $\log p(\mathcal{X}, \mathcal{Y}|\Theta)$ con respecto a los datos desconocidos $\mathcal{Y}$ dados los datos observados $\mathcal{X}$ y la estimación de los parámetros actuales.
Definimos:

$$Q(\Theta, \Theta^{i-1}) = E\left[\log p(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{(i-1)}\right] \tag{6.14}$$

Donde $\Theta^{(i-1)}$ son las estimaciones de los parámetros actuales que usamos para evaluar la esperanza y Θ son los nuevos parámetros que optimizamos para incrementar Q. El lado derecho de la ecuación (6.14) puede ser reescrita como:

$$E\left[\log p(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{(i-1)}\right] = \int_{y \in \mathrm{Y}} \log p(\mathcal{X}, y|\Theta) f(y|\mathcal{X}, \Theta^{(i-1)}) dy.$$

donde $f(y|\mathcal{X}, \Theta^{(i-1)})$ es la distribución marginal de los datos no observados y depende de los datos observados $\mathcal{X}$ y de los parámetros actuales, y Y es el espacio de los valores que toma y.
La evaluación de esta esperanza se llama el paso E del algorítmo.
El paso M del algoritmo EM es maximizar la esperanza que se calculó en el paso 1. Es decir hallar:

$$\Theta^{(i)} = \arg \max_{\Theta} Q(\Theta, \Theta^{(i-1)}).$$

Estos dos pasos se repiten cuantas veces sea necesario. Cada iteración garantiza incrementar la verosimilitud y el algoritmo garantiza la convergencia a un máximo local de la función de verosimilitud.

Máxima verosimilitud de mezclas de densidades via EM

El problema de estimación de mezclas de parámetros de densidades es probablemente una de las aplicaciones más ampliamente usadas del algoritmo EM en la comunidad de reconocimiento de patrones. En nuestro caso suponemos el siguiente modelo probabilístico:

$$p(x|\Theta) = \sum_{i=1}^{M} \alpha_i p_i(x|\theta_i)$$

donde los parámetros son $\Theta = (\alpha_1, \ldots, \alpha_M, \theta_1, \ldots, \theta_M)$ tales que $\sum_{i=1}^{M} \alpha_i = 1$ y cada p_i es una función de densidad parametrizado por θ_i.
La expresión para la log-verosimilitud de los datos incompletos para estas densidades de los datos $\mathcal{X}$ esta dado por:

$$\log(\mathfrak{L}(\Theta|\mathcal{X})) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log \left(\sum_{j=1}^{M} \alpha_j p_j(x_i|\theta_j) \right)$$

que es difícil de optimizar porque contiene el log de la suma. Sin embargo, si postulamos la existencia de los datos no observados $\mathcal{Y} = \{y_i\}_{i=1}^{N}$, la verosimilitud está dada por:

$$\log(\mathfrak{L}(\Theta|\mathcal{X}, \mathcal{Y})) = \log(P(\mathcal{X}, \mathcal{Y}|\Theta)) = \sum_{i=1}^{N} \log(P(x_i|y_i)P(y))$$

$$= \sum_{i=1}^{N} \log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i}))$$

que nos da una forma particular de los componentes de densidad, que pueden ser optimizados usando una variedad de técnicas.
Primero debemos derivar una expresión para la distribución de los datos no observados. Conjeturamos que si $\Theta^g = (\alpha_1^g, \ldots, \alpha_M^g, \theta_1^g, \ldots, \theta_M^g)$ son los parámetros apropiados para la verosimilitud $\mathfrak{L}(\Theta^g|\mathcal{X}, \mathcal{Y})$. Dado Θ^g, se puede calcular $p_j(x_i|\theta_j^g)$ para cada i y j. Usando la regla de Bayes, podemos calcular:

$$p(y_i|x_i, \Theta^g) = \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{p(x_i|\Theta^g)} = \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(x_i|\theta_k^g)}$$

y

$$p(\mathbf{y}|\mathcal{X}, \Theta^g) = \prod_{i=1}^{N} p(y_i|x_i, \Theta^g)$$

donde $\mathbf{y} = (y_1, \ldots, y_N)$ es una instancia de los datos no observados tomados independientemente.
Se puede verificar que la ecuación (6.14) toma la forma:

$$Q(\Theta, \Theta^g) = \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l p_l(x_i|\theta_l)) \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_N=1}^{M} \delta_{l,y_i} \prod_{j=1}^{N} p(y_j|x_j, \Theta^g)$$

la cual se puede reescribir como

$$Q(\Theta, \Theta^g) = \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l) p(l|x_i, \Theta^g) + \sum_{l=1}^{M}\sum_{i=1}^{N} \log(p_l(x_i|\theta_l)) p(l|x_i, \Theta^g).$$

Para maximizar esta expresión, podemos maximizar el término que contiene α_l y el término que contiene θ_l independientemente ya que no estan relacionados.
Para hallar la expresión para α_l, usamos el multiplicador de Lagrange λ con la restricción $\sum_l \alpha_l = 1$, y resolvemos la siguiente ecuación:

$$\frac{\partial}{\partial \alpha_l}\left[\sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l) p(l|x_i, \Theta^g) + \lambda\left(\sum_l \alpha_l - 1\right)\right] = 0$$

$$\Longrightarrow \qquad \sum_{i=1}^{N} \frac{1}{\alpha_l} p(l|x_i, \Theta^g) + \lambda = 0$$

sumando ambos lados sobre l, obtenemos que $\lambda = -N$ resultando:

$$\alpha_l = \frac{1}{N}\sum_{i=1}^{N} p(l|x_i, \Theta^g)$$

Para algunas distribuciones, es posible obtener una expresión analítica para θ_l. Por ejemplo si suponemos distribuciones de componente Gaussiano d-dimensional con media μ y matriz de covarianzas Σ, entonces:

$$p_l(x|\mu_l, \Sigma_l) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_l \mid^{1/2}} e^{-\frac{1}{2}(x-\mu_l)^T \Sigma_l^{-1}(x-\mu_l)}.$$

Usando resultados del cálculo y del algebra matricial se puede demostrar que los estimados de los nuevos parámetros en términos de los parámetros viejos son los siguientes:

$$\alpha_l^{nuevo} = \frac{1}{N}\sum_{i=1}^{N} p(l|x_i, \Theta^g)$$

Figura 6.1: un modelo izquierda-derecha de 4 estados

$$\mu_l^{nuevo} = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^g)}{\sum_{i=1}^{N} p(l|x_i, \Theta^g)}$$

$$\Sigma_l^{nuevo} = \frac{\sum_{i=1}^{N} p(l|x_i, \Theta^g)(x_i - \mu_l^{nuevo})(x_i - \mu_l^{nuevo})^T}{\sum_{i=1}^{N} p(l|x_i, \Theta^g)}$$

Las ecuaciones anteriores desarrollan ambos pasos el de esperanza y el de maximización simultáneamente.

6.4. Tipos de HMM

Un modelo ergódico tiene la propiedad de que cada estado puede ser alcanzado por cualquier otro estado en un número finito de pasos. Por ejemplo para un modelo de 4 estados, este tipo de modelos de cada coeficiente a_{ij} es positivo. Así tenemos

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

Existe un modelo llamado modelo de izquierda-derecha ó modelo Bakis donde la secuencia de estados subyacente asociada con el modelo tiene la propiedad de que al incrementar el tiempo los índices de estado se incrementan (ó se tiene el mismo), i.e., los estados avanzan de izquierda a derecha. Este tipo de modelos tienen la propiedad deseable de que pueden fácilmente modelar las señales cuyas propiedades cambian con el tiempo, por ejemplo el habla. La propiedad fundamental de los modelos de Bakis HMM es que los coeficientes de transición de estado cumplen con

$$a_{ij} = 0, \quad j < i$$

i.e., que las transiciones no se permiten a los estados cuyos índices son menores ó iguales que el estado actual. Las probabilidades de estado inicial tienen la propiedad de que

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

Entonces la sucesión de estados debe empezar en el estado 1 y finalizar en el estado N. Frecuentemente, con los modelos de Bakis, las restricciones adicionales tienen lugar

sobre los coeficientes de transición de estado para hacer más seguro que los cambios grandes en los índices de los estados no ocurran; entonces una restricción de la forma

$$a_{ij} = 0, \quad j > i + \Delta$$

es frecuentemente usada. Sea por ejemplo un modelo de Bakis de 4 estados (Figura6.1) Aquí se tiene un $\Delta = 2$ significa que no se permite saltos de más de 2 estados. La forma de la matriz de transición de estados para éste ejemplo es:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

Para el último estado en un modelo de Bakis, los coeficientes de transición de estados son especificados como

$$a_{NN} = 1$$
$$a_{Ni} = 0, \quad i < N.$$

Se debe observar que a parte del modelo ergódico y del de Bakis, existen otros tantos modelos según muchas variaciones y combinaciones posibles, por ej. un modelo de conexión de 2 modelos paralelos de Bakis. Estos modelos tienen cierta flexibilidad no presente en los modelos de Bakis estrictos.

6.5. Reescalamiento

Consideremos la variable forward $\alpha_t(i)$ definida como

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i \mid \lambda).$$

Se puede ver que $\alpha_t(i)$ consiste de la suma de un número grande de términos, cada uno de la forma

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(O_s) \right),$$

con $q_t = S_i$. Cada término a y b son ≤ 1. Para t suficientemente grande (por ejemplo 100 ó más) el rango dinámico del cálculo de los $\alpha_t(i)$ exedera el rango de precisión de esencialmente cualquier máquina (regularmente en doble precisión). Por lo tanto el único camino razonable de desarrollar el cálculo es incorporando el procedimiento de reescalamiento.

El procedimiento de reescalamiento consiste en multiplicar $\alpha_t(i)$ por un coeficiente de reescalamiento independiente de i(i.e., depende únicamente de t), con el fin de mantener el $\alpha_t(i)$ reescalado dentro del rango dinámico del computador para $1 \leq t \leq T$. Un reescalamiento similar se hace para los coeficientes $\beta_t(i)$ (los cuales también tienden rápidamente a cero exponencialmente) y entonces, en el cálculo final, los coeficientes de reescalamiento se cancelan exactamente.
Para entender mejor el procedimiento de reescalamiento, consideremos la formula de reestimación para los coeficientes de transición de estado a_{ij}. Escribiendo la ecuación de reestimación (6.13) directamente en términos de las variables forward y backward tenemos

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}. \tag{6.15}$$

Consideremos el cálculo de los $\alpha_t(i)$. Para cada t, primero calculamos $\alpha_t(i)$ de acuerdo a la formula de inducción (6.4) , y entonces multiplicamos por un coeficiente de reescalamiento c_t, donde

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$$

El coeficiente reescalado $\hat{\alpha}_t(i)$ se calcula como

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ij} b_j(O_t)}.$$

Ahora calculamos los términos $\beta_t(i)$ de la recursión backward. La única diferencia es que usamos los mismos factores de escala para cada tiempo t para los β's como para las α's. Entonces los betas reescalados son de la forma

$$\hat{\beta}_t(i) = c_t \beta_t(i).$$

Como cada factor de escala efectivamente restaura la magnitud α a valores cercanos a 1, y como las magnitudes de los términos α y β son comparables, el usar los mismos factores de escalamiento para α y β efectivamente se mantienen los cálculos entre cotas razonables.
En términos de las variables reescaladas, tenemos que la ecuación de reestimación (6.15) se puede escribir como

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j)} \tag{6.16}$$

Pero cada $\hat{\alpha}_t(i)$ puede ser escrito como

$$\hat{\alpha}_t(i) = \left[\prod_{s=1}^{t} c_s\right] \alpha_t(i) = C_t \alpha_t(i),$$

y cada $\hat{\beta}_{t+1}(j)$ como

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=t+1}^{T} c_s\right] \beta_{t+1}(j) = D_{t+1} \beta_{t+1}(j).$$

Se tiene que (6.16) puede ser escrito como

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)} \tag{6.17}$$

Los términos $C_t D_{t+1}$ se cancelan fuera del numerador y del denominador de (6.17). El procedimiento de reescalamiento anterior se aplica igualmente a los coeficientes de reestimación de π ó B. El procedimiento de reescalamiento de $\hat{\alpha}_t(i)$ no necesita ser aplicado a cada instante de tiempo t, pero puede ser desarrollado cuando se desee, ó sea necesario (por ej. para prevenir un *underflow*).

6.6. Clasificación basada en HMM

Usando los HMM se construirá un reconocedor de palabras aisladas. Primero consideremos que tenemos un vocabulario de V palabras que serán reconocidas y que cada palabra esta modelada por un HMM distinto. Asumamos también que para cada palabra en el vocabulario, tenemos un conjunto de entrenamiento de K ocurrencias de cada palabra hablada (por uno ó más personas) donde cada ocurrencia de la palabra constituye una sucesión de observaciones, donde tales observaciones son alguna representación apropiada de las características de la palabra (espectral y/ó temporal). Para poder desarrollar nuestro reconocedor de palabras aisladas primero debemos tener en cuenta lo siguiente:

Figura 6.2: Reconocedor de palabras aisladas de un HMM

1. Para cada palabra v en el vocabulario, debemos construir un modelo $\lambda_v = (A_v, B_v, \pi_v)$, i.e., debemos estimar los parametros del modelo que optimize la probabilidad del conjunto de vectores de entrenamiento para la v-ésima palabra.

2. Para cada palabra desconocida, la cual será reconocida, debe realizarse el proceso de la figura 6.2, es decir, determinar la sucesión de observación $O = \{O_1 O_2 \cdots O_T\}$ via un análisis de características cepstrales y continuando con el cálculo de verosimilitudes para todos los posibles modelos, $P(O \mid \lambda_v)$, $1 \leq v \leq V$, seguido por la selección de la palabra cuya verosimilitud sea el más alto, i.e.,
$$v* = \arg \max_{1 \leq v \leq V} [P(O \mid \lambda_v)].$$

3. Los modelos HMM que utilizaremos son del tipo de Bakis como se discutió en la sección 6.4 (véase también la Figura 1.5); por ello la estimación de parámteros basado en el método EM está sujeto a las restricciones
$$\pi = (1, 0, \ldots, 0)^T$$
$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ 0 & a_{22} & a_{23} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & 0 \\ 0 & 0 & \cdots & a_{N-1,N-1} & a_{N-1,N} \\ 0 & 0 & \cdots & \vdots & a_{NN} \end{pmatrix}$$

4. El paso del cálculo de la probabilidad se lleva a cabo usando el algoritmo de Viterbi (i.e., se emplea la trayectoria de máxima probabilidad) y se requiere sobre el orden de VTN^2 calculos.

6.7. Densidades de observación continuas

Hasta ahora se a considerado únicamente el caso cuando las observaciones fueron caracterizadas como símbolos discretos elegidos de un alfabeto finito, y por lo tanto podemos usar una densidad de probabilidad discreta con cada estado de este modelo.

El problema con esta aproximación, al menos para algunas aplicaciones, es que las observaciones son señales continuas. Aunque es posible cuantizar tales señales continuas via libros de códigos (*codebooks*) ésto puede traer una seria degradación asociada con tal quantización. Por lo tanto será una ventaja poder usar los HMM con densidades de observación continua.
Para poder usar una densidad de observación continua, se tienen que hacer algunas restricciones sobre la forma del modelo de la función de densidad de probabilidad (pdf) para asegurar que los parámetros de la pdf pueden ser reestimados en un camino consistente. La representación más general de la pdf, para la cual un procedimiento de reestimación tiene que ser formulado, es una mezcla finita de la forma

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} c_{jm} \mathfrak{N}[\mathbf{O}, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}], \quad 1 \leq j \leq N, \tag{6.18}$$

donde $\mathbf{O} \in \mathbb{R}^O$ es el vector a ser modelado, $c = (c_{jm})_{1 \leq j \leq N, 1 \leq m \leq M}$ es el coeficiente mezclado para la m–ésima mezcla en el estado j y $\mathfrak{N}$ es cualquier función log-concava ó densidad simétrica elíptica. Nosotros tomaremos mezclas de densidades Gaussianas

$$\mathfrak{N}[\mathbf{O}, \boldsymbol{\mu}, \mathbf{U}] = \frac{1}{(2\pi)^{N/2} |\mathbf{U}|^{1/2}} \exp[-\frac{1}{2}(\mathbf{O} - \boldsymbol{\mu})' \mathbf{U}^{-1} (\mathbf{O} - \boldsymbol{\mu})], \tag{6.19}$$

con vector de medias $\boldsymbol{\mu}$ y matriz de covarianza $\mathbf{U}$ para la m–ésima componente mezclado en el estado j. El coeficiente mezclado c_{jm} satisface la restricción estocástica

$$\sum_{m=1}^{M} c_{jm} = 1, \quad 1 \leq j \leq N$$

$$c_{jm} \geq 0, \quad 1 \leq j \leq N \quad 1 \leq m \leq M,$$

de tal forma que la pdf esta correctamente normalizada, i.e.,

$$\int_{-\infty}^{\infty} b_j(\mathbf{x}) d\mathbf{x} = 1, \quad 1 \leq j \leq N$$

La pdf de (6.18) puede ser usada para aproximar, arbitrariamente cerca, cualquier función de densidad continua finita. Esto puede ser aplicado a un amplio rango de problemas.
Las formulas de reestimación para los coeficientes de la densidad mezclada, i.e., $c_{jm}, \boldsymbol{\mu}_{jk}$ y $\mathbf{U}_{jk}$, son de la forma

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)} \tag{6.20}$$

$$\bar{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{O}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.21}$$

$$\overline{\mathbf{U}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\mathbf{O}_t - \boldsymbol{\mu}_{jk})(\mathbf{O}_t - \boldsymbol{\mu}_{jk})^T}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.22}$$

donde T denota la transpuesta y $\gamma_t(j,k)$ es la probabilidad de estar en el estado j al tiempo t con la k-ésima mezcla del componente contado por $\mathbf{O}_t$, ó sea

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{c_{jk}\mathfrak{N}(\mathbf{O}_t, \boldsymbol{\mu}_{jk}, \mathbf{U}_{jk})}{\sum_{m=1}^{M} c_{jm}\mathfrak{N}(\mathbf{O}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm})} \right]$$

El término $\gamma_t(j,k)$ generaliza al $\gamma_t(j)$ de (6.6) en el caso de una mezcla simple, ó una densidad discreta. La formula de reestimación para a_{ij} es idéntica a la usada para densidades de observación discreta. La interpretación de (6.20)a (6.22) es directa.

6.8. El HMM Toolbox

Para la implementación de un reconocedor basado en HMM, se usó el HMM Toolbox escrito por Kevin Murphy (última version: 8 de junio de 1998). Consiste en una serie de rutinas escritas en Matlab que contienen funciones que implementan algunos de los algoritmos que se han mencionado en el capítulo de HMM. El HMM Toolbox permite usar secuencias de observaciones discretas o continuas y en este último de mezclas Gaussianas. Se usaron únicamente las siguientes rutinas del HMM toolbox:

- `mk_stochastic`; construye una matriz estocástica que suma 1 por renglones.
- `mk_leftright_transmat`; construye una matriz estocástica de tipo Bakis. En el método EM las distintas iteraciones mantienen esta restricción.
- `mhmm_em`; función principal del HMM Toolbox que estima los parámetros del HMM por el método EM.
- `mhmm_logprob`; calcula las probabilidades de observación dado el modelo.

Para la estimación inicial de las mezclas Gaussianas se usa la función
`gmmdistribution.fit`
de Matlab. Análogamente la distribución del estado inicial se tomó como $\pi = (1, 0, \ldots, 0)^T$. El método EM preserva esta restricción. La guía del autor del HMM Toolbox se puede encontrar en:
`http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm_usage.html`
El HMM Toolbox y otro software elaborado por el autor se pueden descargar en:
`http://www.cs.ubc.ca/~murphyk/Software/index.html`

Capítulo 7

Resultados

7.1. Descripción de los CORPORA

Se usaron cuatro bases de datos (CORPORA) A,B,C,D para validar la metodología de reconocimiento de voz usando DTW. Cada corpus consistió de 10 grabaciones por la misma persona de los dígitos del 0 al 9 en formato wav, con frecuencias de muestreo de 8000 Hz. Cada corpus incluía tambien 10 grabaciones extra de 10 dígitos grabados al azar. Con las primeras 100 grabaciones se generaron las secuencias prototipo y con las 10 extra las secuencias por clasificar. La codificación que se uso fué: `X_YY.wav`, donde `X` 0:9 corresponde al dígito y `YY` 1:10 corresponde a la instancia.

El dispositivo de grabación de corpus A fué un dispositivo MP3 con frecuencia de muestreo de 8000 Hz, se grabaron las 10 instancias de cada dígito en un mismo archivo repitiendo la voz, p.ej., "uno", "uno",..,"uno" (10 veces). Posteriormente se editaron con el software "Audacity" de distribución libre, segmentando manualmente los tramos de voz que correspondían a cada instancia y exportando en archivos wav separados.

Los CORPORA B, C fueron hechos por grupos de alumnos del curso de Taller de Modelado II. Se les hizo la recomendación de grabar separadamente las 10 instancias de cada dígito para evitar posibles correlaciones involuntarias al grabar de manera continuada. Fué detectado, por ejemplo, que en el corpus A, la última instancia tendía a ser más modulada, posiblemente por el modo de hablar defeño de "cantar" los finales de palabra. Todas las grabaciones fueron hechas con los mismos dispositivos MP3 con los que se generó el corpus A, pero siempre grabados por la misma persona; el corpus B por una persona del sexo femenino y el corpus C por una persona del sexo masculino. Los extremos de voz se detectaron manualmente con el Audacity.

Se usó también el corpus E, del proyecto prodivoz de la Unversidad Nacional de Rosario, Argentina. formado por 750 archivos grabados por 25 locutores de ambos sexos, con tres instancias de cada dígito del 0 al 9: 25*3*10=750, cada grabación en formato wav con una frecuencia de muestreo de 11025 Hz. La grabación estaba identificada por el nombre del archivo codificado como `XYYZ.wav` donde `X` 0:9 corresponde al dígito, `YY` 1:25 al locutor y `Z` 1:3 a la instancia. No se disponía información del tipo de dispositivo usado para las grabaciones, pero dada la alta frecuencia de muestreo, se puede suponer que eran de buena calidad.

7.2. Corpus A

El reconocimiento se hizo con una versión de Mathematica de los módulos de preprocesamiento y de clasificación. El tiempo aproximado para generar las secuencias de vectores patrón fué aproximadamente de 4 horas en una PC con un procesador Athlon y 512 Mb en RAM. El reconocimiento de 10 dígitos desconocidos fué de una hora aproximadamente. Una revisión posterior de los archivos de sonido reveló algunos errores en la segmentación manual de las instancias de los dígitos, pues algunos fueron cortados prematuramente. Se obtuvo un porcentaje de reconocimiento del 70 %.

7.3. CORPORA B, C, D

Cada instancia se grabó por separado. El código se hizo en Matlab. El promedio de tiempo al generar las secuencias de vectores de atributos para las 100 instancias fué del orden de 10 minutos. La clasificación duró del orden de 5 minutos. Se obtuvo un porcentaje de reconocimiento del 90 %.

7.4. Corpus E, reconocimiento de dígitos dentro del corpus

Se particionaron los 750 archivos en 100 para de validación y 650 instancias de dígitos patrón. Se repitió el procedimiento unas tres veces con resultados muy similares de reconocimiento que iban del 99 % al 100 %.

7.5. Corpus E', reconocimiento de dígitos fuera del corpus

Se usaron los 750 archivos como instancias de dígitos patrón. Se grabaron 10 instancias de cada uno de los 10 dígitos separadamente a 8000 Hz con un dispositivo MP3. Posterioremente se hizo el *upgrade* a 11025 Hz para hacer compatibles los datos de prueba con los datos patrón. Se usó software libre SoX (Sound Exchange). El porcentaje de éxito fué del 78 %.

7.6. Resultados de la clasificación usando DTW

Se implementó el reconocimiento de dígitos aislados implementando directamente el código del procedimiento descrito en el capitulo 6 , primeramente en Mathematica (corpus A) y posteriormente se migró a Matlab.

Corpus	No. de archivos (patrón+prueba)	Frecuencia	Porcentaje de éxito
A	100+10	8000	70 %
B	100+10	8000	90 %
C	100+10	8000	90 %
D	100+10	8000	90 %
E	750	11025	99.3–100 %
E'	750	11025	78 %

En el Apéndice E se muestra el listado de los directorios que contienen los corpora D y E, así como algunas salidas de corridas usando la implementación del algoritmo DTW en Matlab del Apéndice C.

7.7. Resultados de la clasificación usando HMM

Se usaron algunas rutinas del HMM toolbox, tal como se describe en las sección 6.8. Se trabajó con los corpora C y E variando el número de estados N y el número de mezclas Gaussianas M.

Para el corpus C se obuvieron los siguiente porcentajes promedio de reconocimiento realizando 50 corridas para cada caso:

No. de estados y de mezclas	Promedio de éxitos en %
$N = 3,\ M = 1$	67.40 %
$N = 4,\ M = 1$	64.20 %
$N = 5,\ M = 1$	58.40 %
$N = 3,\ M = 2$	52.40 %
$N = 4,\ M = 2$	43.80 %
$N = 5,\ M = 2$	38.00 %

El mayor porcentaje de éxitos (67.40 %) se logra con $N = 3$ y $M = 1$.
Para el corpus E se obuvieron los siguiente porcentajes promedio de reconocimiento realizando 10 corridas para cada caso.

No. de estados y de mezclas	Promedio de éxitos en %
$N = 3,\ M = 2$	87.89 %
$N = 4,\ M = 1$	90.53 %
$N = 4,\ M = 2$	88.95 %
$N = 5,\ M = 2$	90.00 %

El mayor porcentaje de éxitos (90.53 %) se logra con $N = 4$ y $M = 1$, el modelo de Markov oculto para este caso lo podemos visualizar con su matriz de transición correspondiente $A = (a_{ij})$

$$\mathbf{A} = \begin{bmatrix} 0.9031 & 0.0969 & 0 & 0 \\ 0 & 0.8819 & 0.1181 & 0 \\ 0 & 0 & 0.9884 & 0.0116 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

La matriz (4x1) de mezclas Gaussianas

$$c = \begin{bmatrix} 0.1161 \\ 0.2713 \\ 0.4741 \\ 0.1384 \end{bmatrix},$$

y la matriz (4x1) $\pi = (\pi_i)$.

$$\pi = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

es la distribución de probabilidades del estado inicial.

Capítulo 8

Conclusiones y perspectivas

Se implementó el reconocimiento de dígitos aislados con dos metodologías: DTW y HMM. Para la primera se usaron seis bases de datos (corpora), denominados A, B, C, D, E y E' en este trabajo, en forma de archivos grabados en formato *.wav. Los corpora A–D estaban formados por 100 archivos con 10 instancias de grabación de cada dígito del 0 al 9 más 10 archivos de dígitos seleccionados al azar. Estos corpora fueron grabados por las mismas personas y la frecuencia de muestreo fue de 8000 Hz. El corpora A, el primer en grabarse, mostró inconsistencias en la terminación e inicio de la grabación y fue el que mostró los resultados más pobres (70 %). No se implementaron algoritmos de detección de extremos (*end point detection*). Los corpora B-D fueron grabados teniendo cuidado en el inicio y el término de cada grabación y cada instancia de grabación fue hecha por separado. Los resultados de reconocimiento mejoraron hasta un 90 %. El corpora E consistió de 750 grabaciones de la Universidad Nacional del Rosario, Argentina grabadas a 11025 Hz de tres instancias del mismo dígito grabados por la misma persona y un total de 25 locutores de distinto género (3*25*10=750). El reconocimiento con el corpora E se hizo seleccionando 100 grabaciones al azar para el reconocimiento y 650 como dígitos patrón; se obtuvo la mayor tasa de reconocimiento que varió entre 99.3 %–100 %. La explicación que damos es que el corpora E fue grabado con un mejor equipo y con una frecuencia de muestreo más alta. El reconocimiento con el corpora E' consistió en usar las 750 instancias de grabación como dígitos patrón y 100 grabaciones hechas con los mismos parámetros que los corpora B–D. Para poder hacer la comparación con los dígitos patrón se hizo el *upgrade* de 8000 Hz a 11025 Hz utilizando software libre SoX (Sound Exchange). En este caso el procentaje de reconocimiento llegó a un 78 %. Como es natural la caida en el porcentaje de reconocimiento se debe a que los datos de reconocimiento no pertenecían al corpus original, además la calidad de los datos a

reconocer. Todos los corpora, excepto los de la UNR, fueron grabados con dispositivos MP3 o micrófonos simples conectados a la computadora, pues no se cuenta con un laboratorio de grabación.
Se implementó el reconocimiento usando HMM con los corpora C y E. Los corpora A–B tuvieron una calidad muy baja para poder implementar esta metodología, pues no se lograba la convergencia del método EM. La convergencia del algoritmo EM se logró con el corpora C imponiendo las restricciones del modelo de Bakis y de la distribución de probabilidad del estado inicial y estandarizando los datos de entrenamiento. Para el corpora C se obtuvo el mejor modelo HMM con $N = 3$ estados y $M = 1$ mezclas Gaussianas con un porcentaje de reconocimiento del 67.40 %. Para el corpora E se obtuvo el mejor modelo con $N = 4$, $M = 1$ con un 90.53 % de reconocimiento.
En conclusión se encontraron mejores resultados en el corpora E usando DTW que en HMM, sin embargo la metodología HMM es un método de reconocimiento de patrones basado en la comparación directa de una muestra con toda la base patrón que por lo tanto no puede implementarse para bases de datos grandes. La metodología HMM aunque ofreció menor porcentaje de reconocimiento en el mismo corpus, puede admitir una base de datos arbitraria y al contrario de lo que sucede con los métodos de reconocimiento de patrones, entre mayor sea el tamaño de la base de dígitos patrón mejor estimación se tiene de los parámetros. La debilidad de la metodología HMM, como en cualquier método de optimización (en este caso el método EM) es que se necesita una aproximación inicial buena para lograr la convergencia y aún así no está garantizado el máximo global de la función de verosimilitud.
Las perpectivas que se ofrecen a partir de este trabajo en relación al reconocimiento de dígitos aislados están:

- implementaciones libres de la metodología HMM usando software libre tal como HTK o software comercial como SAPI, entre otros.
- Implementar directamente el método EM o Baum–Welch directamente en Matlab. La razón es que el entrenamiento de modelos de Bakis requieren del entrenamiento con secuencias multiples de entrenamiento [11] que no está implementada en el HMM Toolbox.
- Implementar detección de extremos.
- Implementar métodos basado en algoritmos genéticos para buscar máximos globales de la función de verosimilitud.

Bibliografía

[1] R.E. Bellman (1962). Applied dynamic programming. Princeton University Press.

[2] R.Berkow, M.D. (1997). *Manual Merck de información médica para el hogar*. Océano. pp. 1059–1062.

[3] Bilmes, J.A. (1998). "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models". Reporte. International Computer Science Institute, Berkeley, Ca.

[4] J.W. Cooley, P. Lewis y P. Welch, (1969). "The Fast Fourier Transform and its Applications", IEEE Trans. on Education **12**, 1,28-34.

[5] Hsu, Hwei P. (1983). *Análisis de Fourier* Addison-Wesley Iberoamericana.

[6] B. Kedem (1994). *Series analysis by higher order crossings* IEEE Press.

[7] Meyer, P. L. (1992). *Introductory probability and statistical applications*. Addison-Wesley Iberoamericana.

[8] LinkLipschutz, S. (1971). *Teoría y problemas de probabilidad*, McGraw-Hill.

[9] P.E. Pfeiffer (1990). *Probability for Applications.* Springer-Verlag.

[10] B. Plannerer (2005). "An introduction to speech recognition." Notas no publicadas.

[11] L.R. Rabiner (1989). "A tutorial on hidden Markov models and selected applications in speech Recognition." Proceedings of the IEEE **77**, No.2, pp. 257-286.

[12] L.R. Rabiner y M.R.Sambur (1975). "An algorithm for determining the endpoints of isolated utterances." Bell System Tech. Journal, 54:297–35, Feb. 1975.

[13] L.R. Rabiner y Juang, Biing-Hwang (1993). *Fundamentals of Speech Recognition*. Englewoods Cliffs, N.J. Prentice-Hall.

[14] S.S. Stevens, J.Volkmann y E.B.Newman (1937). "A Scale for the Measurement of the Psychological Magnitude Pitch." The Journal of the Acoustic Society of America. pp. 185-190.

[15] L.R. Rabiner y R.W. Schafer (1978). *Digital Processing of Speech Signals*. Prentice Hall,Englewood Cliffs, NJ.

[16] H. Sakoe y S. Chiba (1978). "Dynamic Programming Algorithm Optimization for Spoken Word Recognition." IEEE Trans. Acoustics, Speech, Signal Proc., pp. 43–49.

[17] Prodivoz: Laboratorio de procesamiento de voz de la Universidad Nacional del Rosario, Argentina. `www.fceia.unr.edu.ar/prodivoz`

[18] K. Murphy. HMM toolbox para Matlab.

`http://people.cs.ubc.ca/~murphyk/`.

[19] K. Murphy. "How to use the HMM toolbox".

`http://people.cs.ubc.ca/~murphyk/Software/HMM/hmm_usage.html`

Apéndice A

La DTFT y la DFT

La transformada de Fourier en tiempo discreto (DTFT) $X(\omega)$ de una señal $x(k)$, $k = -\infty, \ldots, \infty$ se define como

$$X(\omega) = \sum_{k=-\infty}^{k=+\infty} x(k) e^{-j\omega k}$$

y su inversa es,

$$x(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{j\omega k} \, d\omega$$

La DTFT $X(\omega)$ toma valores complejos y es una función continua y periódica en ω. El periodo 2π, se representa normalmente en el intervalo $[-\pi, \pi]$. Al evaluar numéricamente la DTFT se presentan dos problemas:

a) La secuencia $x(k)$ puede consistir de un numero infinito de puntos. La convergencia de la serie significa que

$$\sum_{k=0}^{\infty} |x(k)| < \infty \quad \text{y} \quad \sum_{k=-\infty}^{0} |x(k)| < \infty$$

es decir $(x(k))_{k=0}^{\infty}$ y $(x(k))_{k=-\infty}^{0}$ deberán ser funciones en ℓ_1. Para un sistema LTI (lineal invariante en el tiempo) existe L tal que $x(k+L) = x(k)$, para toda k y la serie $(x(k))_{k=0}^{\infty}$ no pertenece a ℓ_1 ya que la suma diverge.

b) $X(\omega)$ es una función continua de la frecuencia ω y debe ser discretizada para trabajar en un procesador digital.

Para resolver el primer problema consideraremos que la secuencia de entrada esta formada por un vector de L puntos siendo 0 para los valores fuera del rango $[0, L]$. Para el segundo, consideraremos que $X(\omega)$ se evalúa en un número finito N de frecuencias equidistantes en el intervalo $[-\pi, \pi]$ con incrementos de $2\pi/N$, es decir se consideran el conjunto discreto de frecuencias $w_n = 2\pi n/N$ con $n = 0, 1, \ldots, N-1$. Si se elige N lo suficientemente grande, los valores $X(2\pi n/N)$ se aproximan a la función continua $X(\omega)$. Al muestrear la DTFT de esta manera se obtiene la expresión correspondiente a la trasformada discreta de Fourier (DFT) que en MATLAB se implementa mediante el algoritmo conocido como FFT (Fast Fourier Transform). Para evitar problemas de muestreo insuficiente se debe elegir N tal que $N > L$. Ver [4]y[5]

Apéndice B

Programa para preprocesamiento

```
function prep
% Construye las secuencias de vectores de caracter\'i sticas

disp(['Creando .feat para cada archivo de prueba']);
    listFiles=dir(['digitos-prueba-11025Hz\*.wav']);
    for i=1:length(listFiles)
        soundIN=wavread(['digitos-prueba-11025Hz\' listFiles(i).name]);
        X=SequenceFeatureVector(soundIN, 25, 10, 11025, 12, 14);
        cadena=listFiles(i).name;
        t=length(cadena);
        cadena=cadena(1:t-4);
        save(['feat-patron\' cadena '.mat'], 'X');
        disp(['Done File: ' listFiles(i).name]);
    end



function X=SequenceFeatureVector(soundfile,lengthWindowMs,shiftWindowMs,..
                        sampleRate,noMelCoefs,noChannels)
% Calcula la secuencia de vectores de atributos para la se\~nal soundfile

% Entradas
%      soundfile -> archivo de sonido (ya como un vector)
```

```
% lengthWindowMs -> Tama\~no de la ventana en ms
%  shiftWindowMs -> corrimiento de la ventana en ms
%     sampleRate -> Frecuencia de muestreo
%     noMelCoefs -> n\'umero de coeficientes de Mel
%     noChannels -> n\'umero de canales
%
% Salidas
% X ->  matriz X de atributos de dimension
% nwin(numero de ventanas)x(noMelCoefs+1)
% cada fila contiene los coeficientes de mel
% mas energia de la ventana


[L,wlength,wshift,nwin]= ...
   ParametersOfSignal(soundfile,lengthWindowMs,shiftWindowMs,sampleRate);

senf=enfasis(soundfile');

for window=1:nwin
    offset=(window-1)*wshift+1;
    spect=SFT(senf,offset,wlength);
    h=melMatrix(wlength, noChannels, sampleRate);
    mfccs=MFCC(noMelCoefs, noChannels, wlength, spect, sampleRate);
    e=energy(soundfile(offset:offset+wlength));

    if window==1
        dtminus=0;
    else
        dtminus=2;
    end

    if window==nwin
        dtplus=0;
    else
        dtplus=2;
    end

 deltac=...
```

```
 dc(senf, offset, dtplus, dtminus, noMelCoefs, ...
       noChannels, wlength, sampleRate);

 delta2c=...
 ddc(senf, offset, dtplus, dtminus, noMelCoefs, ...
          noChannels, wlength, sampleRate);

    X(window,:)=[mfccs,deltac,delta2c,log(e)];
end



function [L,wlength,wshift,nwin]= ...
  ParametersOfSignal(soundfile,windowlengthMs,shiftMs,samplefrequency)

% Calcula par\'ametros de la se\~nal de sonido
%
% Entradas
% soundfile -> archivo de sonido (ya como un vector)
% windowlengthMs -> tama\~no de la ventana (t\'i picamente entre 16  y 25 ms)
% shiftMs -> cambio de la ventana o avance (t\'i picamente 10 ms)
% samplefrequency -> Frecuencia de muestreo
%
% Salidas
% L -> Longitud de la se\~nal
% wlength -> ancho de ventana (entero)
% wshift -> shift (entero)
% nwin -> n\'umero de ventanas


dt=1/samplefrequency;                    % intervalo de muestreo
L=length(soundfile);                     % longitud de la se\~nal
DelthaT=L*dt;                            % duraci\'on de la se\~nal
wlength=ceil(windowlengthMs*10^(-3)/dt); % aseguramos tener un entero
wshift=ceil(shiftMs*10^(-3)/dt);
nwin=floor((L-wlength)/wshift);
```

```
function senf=enfasis(soundfile)

% Elimina el decaimiento de -6 dB por octava
%
% Entradas
% soundfile -> archivo de sonido (ya como un vector)
%
% Salidas
% senf -> archivo de sonido enfatizado


smin=min(soundfile);
smax=max(soundfile);
std=-1+2*(soundfile-smin)/(smax-smin);
stda=std(1:length(std)-1);
stda=[std(length(std)),stda];
senf=std-0.97*stda;
senf=senf(2:length(senf));


function V=SFT(signal, offset, wlength)

% Calcula el espectro de potencia de signal para una ventana
% de longitud wlength que comienza en offset
%
% Entradas
% signal -> Vector que contiene una se\~nal enfatizada
% offset_Integer -> Entero que representa donde comienza la ventana
% wlength_Integer -> Longitud de la ventana
%
% Salidas
%
%   V -> Espectro de potencia
```

```
L=length(signal);
if offset+wlength>L
   disp('Error');
   V=0;
else
   k=1:wlength;
   w=0.54-0.46*cos(2*pi*k/(wlength-1));    %Hamming Window
   v=signal(offset+k).*w;

   %calculamos la transformada discreta de Fourier
   V=abs(fft(v)).^2;
end
```

```
function c=MFCC(noMelCoefs,noChannels,lengthWindow,...
      powerSpectrum,sampleRate)

% Calcula los coeficientes cepstrales de Mel
%
% Entradas
%  noMelCoefs -> n\'umero de coeficientes de Mel
%  noChannels -> n\'umero de canales
%  lengthWindow -> Tama\~no de la ventana (calculada por ParameterOfSignal)
%  powerSpectrum -> Espectro de potencia (calculado por SFT)
%  sampleRate -> frecuencia de muestreo
%
% Salidas
%  c -> Mel Frequency Cepstral Coeficients (es un vector)
%


halfLengthWindow=floor(lengthWindow/2+1);
halfPowerSpectrum=powerSpectrum(1:halfLengthWindow);
n=melMatrix(lengthWindow, noChannels, sampleRate);
G=(n*halfPowerSpectrum')';
```

```
c=zeros(1,noMelCoefs);

k=0:noChannels-1;
 for q=1:noMelCoefs
     c(q)=sum(log(abs(G(k+1))).*cos(pi*q*(2*k+1)./(2*noChannels)));
 end




function n=melMatrix(lengthWindow, noChannels, sampleRate)

% Calcula la matriz de coeficientes de Mel
%
% Entradas
% lengthWindow -> Tama\~no de la ventana (calculada por ParameterOfSignal)
% noChannels -> N\'umero de canales
% sampleRate -> Frecuencia de muestreo
%
% Salidas
% n -> matriz de coeficientes de Mel
%


halfLengthWindow=floor(lengthWindow/2+1);

melmax=2595*log(1+(sampleRate/2)/700);
dmel=melmax/(noChannels+1);
df=sampleRate/lengthWindow;

n=zeros(noChannels,halfLengthWindow);

for k=1:noChannels
    ns=nc(k-1,dmel,df);
    while ns<=nc(k,dmel,df)
        n(k,ns)=(ns-nc(k-1,dmel,df))/(nc(k,dmel,df)-nc(k-1,dmel,df));
        ns=ns+1;
    end
```

```
    while ns<=nc(k+1,dmel,df)& ns<=halfLengthWindow
        n(k,ns)=(ns-nc(k+1,dmel,df))/(nc(k,dmel,df)-nc(k+1,dmel,df));
        ns=ns+1;
    end
end

function nc=nc(k,dmel,df)
fc=(k+1)*dmel;   %Mel centers
freq=700*(-1+exp(fc/2595));

nc=ceil(freq/df);


function e=energy(signal)

% Calcula la energia de la se\~nal
%
% Entrada
% signal -> la se\~nal
%
% Salida
%  e-> energia de la se\~nal

e=dot(signal,signal);




function dlc=dc(signal, offset, tau_plus, tau_minus, ...
            noMelCoefs, noChannels, lengthWindow, sampleRate)

% Calcula la derivada de c=MFCC
%
% Entradas
% signal -> Vector que contiene una se\~nal enfatizada
% offset -> Entero que representa donde comienza la ventana
% tau_plus y tau_minus -> intervalo de tiempo usalmente 2<=tau<=4 es usado
```

```
% noMelCoefs -> n\'umero de coeficientes de Mel
% noChannels -> n\'umero de canales
% lengthWindow -> Tama\~no de la ventana (calculada por ParameterOfSignal)
% sampleRate -> frecuencia de muestreo
%
% Salidas
%  dlc -> la derivada de c=MFCC
%


spectrumplus=SFT(signal, offset+tau_plus, lengthWindow);
spectrumminus=SFT(signal, offset-tau_minus, lengthWindow);
mfccplus=MFCC(noMelCoefs, noChannels, lengthWindow,...
 spectrumplus, sampleRate);
mfccminus=MFCC(noMelCoefs, noChannels, lengthWindow,...
 spectrumminus, sampleRate);

dlc=mfccplus-mfccminus;

function dldlc=...
 ddc(signal, offset, tau_plus, tau_minus, ...
       noMelCoefs, noChannels, lengthWindow, sampleRate)

% Calcula la segunda derivada de c=MFCC
%
% Entradas
% signal -> Vector que contiene una se\~nal enfatizada
% offset -> Entero que representa donde comienza la ventana
% tau_plus y tau_minus -> intervalo de tiempo usalmente 2<=tau<=4 es usado
% noMelCoefs -> n\'umero de coeficientes de Mel
% noChannels -> n\'umero de canales
% lengthWindow -> Tama\~no de la ventana (calculada por ParameterOfSignal)
% sampleRate -> frecuencia de muestreo
%
% Salidas
%  dldlc -> la derivada de c=MFCC
%
```

```
dldlc=dc(signal, offset+tau_plus, tau_plus, tau_minus,...
              noMelCoefs, noChannels, lengthWindow, sampleRate)-...
    dc(signal, offset-tau_minus, tau_plus, tau_minus,...
             noMelCoefs, noChannels, lengthWindow, sampleRate);
```

Apéndice C

Programa de clasificación usando DTW

```
function Clasificacion

% Clasifica los archivos .mat para
% digitos prueba y patr\'on

% C.mat contiene la matriz de varianzas y covarianzas
% Para la distancia de Mahalanobis gamma = G;
% Para la distancia pesada gamma = diag(G)


global gamma

load('cov_digitos_11025\C.mat');
gamma=C;

%Leemos todos los archivos de prueba
listFilesTest=dir(['feat_digitos_8000Hz\*.mat']);

%Leemos todos los archivos patr\'on
listFilesPatron=dir(['feat_11025Hz\*.mat']);


disp(['Clasificando las pruebas con archivos patr\'on']);
```

```
exitos=0;
cont=0;
for i=1:length(listFilesTest)

    load(['feat_digitos_8000Hz\' listFilesTest(i).name]);
    Xtest=X;

      disp(['cargo d\'i gito de prueba ' listFilesTest(i).name]);
    distances=[];

    for j=1:length(listFilesPatron)
      load(['feat_11025Hz\' listFilesPatron(j).name]); % Carga una matriz W
      Xpatron=X;

      clear X;
        distances=[distances,distance(Xpatron,Xtest)];
    end
       [valor,pos]=min(distances);
    digit=listFilesPatron(pos).name(1);

 disp(['El archivo  ' listFilesTest(i).name '  es un ...
    ' digit ' dist.minima se alcanze en ' listFilesPatron(pos).name ]);
    cont=cont+1;
% Cuenta el numero de \'exitos
     if listFilesTest(i).name(1) == digit,
       exitos=exitos+1;
     end
disp(['Se lleva un porcentaje de '  ...
                   num2str(100.0*exitos/cont) ' % de  exitos']);
 end

disp(['Se obtuvo un porcentaje de '  ...
     num2str(100.0*exitos/length(listFilesTest)) ' % de  exitos']);


function dst=distance(W, X)
```

```
% Funci\'on para hacer la comparaci\'on entre dos secuencias de vectores
% asociados a dos se\~nales diferentes, esta comparaci\'on es realizada por
% medio del procedimiento DTW (Dynamic Time Warping).
%
% Entradas
% W, X  -> secuencias de vectores de caracter\'\i  sticas como una
% matriz
%
% Salidas
%  Dst -> distancia entre esos dos vectores
%


sw=size(W); Tw=sw(1);
sx=size(X); Tx=sx(1);

phi(1,1,:)=[0,0];
delta(1,1)=d(W(1,:),X(1,:));

for i=2:Tw
    delta(i,1)=d(W(i,:),X(1,:))+delta(i-1,1);
    phi(i,1,:)=[i-1,1];
end

for j=2:Tx
    delta(1,j)=d(W(1,:),X(j,:))+delta(1,j-1);
    phi(1,j,:)=[1,j-1];
    for i=2:Tw
        ds=d(W(i,:),X(j,:));
        %No se usa el argumento 2 como en el Plannerer
        pds1=delta(i,j-1)+ds;
        pds2=delta(i-1,j-1)+ds;
        pds3=delta(i-1,j)+ds;

        min=pds1; select=[i,j-1];

        if pds2<min
            min=pds2;
```

```
            select=[i-1,j-1];
        end

        if pds3<min
            min=pds3;
            select=[i-1,j];
        end

        delta(i,j)=min;
        phi(i,j,:)=select;
    end
end

dst=delta(Tw,Tx);



function dis=d(Wv,Xv)
global gamma;

%Funci\'on que calcula la distancia entre dos vectores de caracter\'isticas

U=Wv-Xv;
dis = U*gamma*U';
```

Apéndice D

Programa de clasificación usando el HMM Toolbox

```
clear all;
clc;
Wall=[];
for digit=0:9
  load(['Feat\' num2str(digit) '.mat']);
  Wall=[Wall W];
end
m=mean(Wall,2);
s=std(Wall,0,2);
clear Wall W X;

O=37;
Q=2;
M=2;
for digit=0:9
    load(['Longitudes\' num2str(digit) 'long.mat']);
    load(['Feat\' num2str(digit) '.mat']);
    T=size(W,2);
    mrep=repmat(m,[1 T]);
    srep=repmat(s,[1 T]);
    %Wtilde=W;
    %Wtilde=W./mrep;
    Wtilde=(W-mrep)./srep;
```

```
for i=1:10
  datacell{1,1,i}=Wtilde(:,longitudes(1,i)+1:longitudes(1,i+1));
end
prior0 = normalise(rand(Q,1));
transmat0 = mk_leftright_transmat(Q, rand);
flag = 0; %Repetir la estimacion si el loglkd es -inf
while flag == 0
  disp(['still in digit                ' num2str(digit)]);
  obj = gmdistribution.fit(Wtilde',Q*M,'CovType',...
  'diagonal','Regularize',.02);
  mu0=obj.mu;
  mu0=mu0';
  Sigma0=obj.Sigma;
  SigmaM=diag(Sigma0(:,:,1));  %<- In case covariance matrix is diagonal
  for i=2:Q*M
    SigmaM(:,:,i)=diag(Sigma0(:,:,i));
end;
Sigma0=SigmaM;
mu0 = reshape(mu0, [O Q M]);
Sigma0 = reshape(Sigma0, [O O Q M]);
mixmat0 = mk_stochastic(rand(Q,M));
[LL, prior1, transmat1, mu1, Sigma1, mixmat1] = ...
mhmm_em(datacell, prior0, transmat0, mu0,Sigma0, mixmat0,...
 'verbose',0,'max_iter', 10);
if LL(end)~= -Inf
   flag =1;
end
end
if digit == 0
    prior=prior1;
    transmat=transmat1;
    means=mu1;
    Sigma=Sigma1;
    mixmat=mixmat1;
end
prior(:,:,digit+1)=prior1;
transmat(:,:,digit+1)=transmat1;
means(:,:,:,digit+1)=mu1;
```

```
    Sigma(:,:,:,:,digit+1)=Sigma1;
    mixmat(:,:,digit+1)=mixmat1;
end

%Prepare for recognition
listfiles=dir (['FeatTest\*.mat']);
for i=1:length(listfiles)
     load(['FeatTest\' listfiles(i).name]);
     X=X';
     mrep=repmat(m,[1 size(X,2)]);
     srep=repmat(s,[1 size(X,2)]);
     %X=X./mrep;
     X=(X-mrep)./srep;
  for digit=1:10
     loglik(digit) = ...
     mhmm_logprob(X, prior(:,:,digit), transmat(:,:,digit),...
     means(:,:,:,digit), Sigma(:,:,:,:,digit), mixmat(:,:,digit));
  end
  [c,num]= max(loglik);
  disp(num2str(num-1));
end
```

Apéndice E

Ejemplos de corridas usando DTW

Figura E.1: Directorio del corpora D de archivos wav

Figura E.2: Detalle del corpora E de archivos wav de la UNR, Argentina.

Figura E.3: Resultados corpora A

Figura E.4: Resultados corpora E, última página

Figura E.5: Resultados corpora E', primer página

Figura E.6: Resultados corpora E', segunda página

Figura E.7: Resultados corpora E', tercer página

Figura E.8: Resultados corpora E', cuarta página

Figura E.9: Resultados corpora E', última página

Apéndice F

Programa de reconocimiento del corpora C con HMM

```
clear all;
clc;

Wall=[];

%Se calcula la media y desviacion estandar de todas las muestras
for digit=0:9
   load(['Feat\' num2str(digit) '.mat']);
   W=X;
   Wall=[Wall W];
end
  m=mean(W,2);
  s=std(W,0,2);
  clear Wall W X;


O=37;
Q=3;
M=1;
left_right=0;
```

```
for digit=0:9
 load(['Longitudes\' num2str(digit) 'long.mat']);
 load(['Feat\' num2str(digit) '.mat']);
 W=X;
 T=size(W,2);
 mrep=repmat(m,[1 T]);
 srep=repmat(s,[1 T]);

%Se normalizan los vectores de caracteristicas en R^O

 Wtilde=(W-mrep)./srep;


%Los datos de entrenamiento se guardan en arreglos de celdas
    for i=1:10
         datacell{1,1,i}=Wtilde(:,longitudes(1,i)+1:longitudes(1,i+1));
    end

%Se inicializa el vector de probabilidad incial aleatoriamente
    prior0 = normalise(rand(Q,1));
%Modelo de Bakis
    transmat0 = mk_leftright_transmat(Q, rand);

%Se estima 10 veces cada modelo de Markov. Se seleccionan solo aquellos
%que arrojen log-verosimilitud finita y se toma el que tenga maxima
%verosimilitud
    for iter=1:10

      flag = 0;
    while flag == 0
    disp(['still in digit                ' num2str(digit)]);
    j=ceil(10.*rand(1,1));
    [mu0, Sigma0] = mixgauss_init(Q*M,datacell{j}, 'diag','rnd');
    mymu0=mu0;
    mySigma0=Sigma0;
    obj = gmdistribution.fit(Wtilde',Q*M,'CovType',...
    'diagonal','Regularize',.005);
    mu0=obj.mu;
```

```
    mu0=mu0';
    Sigma0=obj.Sigma;

    SigmaM=diag(Sigma0(:,:,1));
    for i=2:Q*M
        SigmaM(:,:,i)=diag(Sigma0(:,:,i));
    end;
    Sigma0=SigmaM;

    mu0 = reshape(mu0, [O Q M]);
    Sigma0 = reshape(Sigma0, [O O Q M]);
    mixmat0 = mk_stochastic(rand(Q,M));

    [LL, prior1, transmat1, mu1, Sigma1, mixmat1] = ...
    mhmm_em(datacell, prior0, transmat0, mu0,Sigma0, mixmat0,...
     'verbose',0,'max_iter', 10);
    if LL(end)~= -Inf
       flag =1;
    end


    end

       params{iter}=  {LL(end), prior1, transmat1, mu1, Sigma1, mixmat1};



    end

%Se extraeen las log-verosimilitudes de cada iteracion
    lkhd=params{1}{1};
    for k=2:10
        lkhd=[lkhd params{k}{1}];
    end
 %Se extrae el indice que corresponda a la maxima verosimilitud
     [mxlkd,ind]= max(lkhd);

 %Se extraen los parametris del HMM
```

```
     for l=1:6

     LL= params{ind}{1};
     prior1 = params{ind}{2};
     transmat1 = params{ind}{3};
     mu1= params{ind}{4};
     Sigma1 = params{ind}{5};
      mixmat1 = params{ind}{6};
     end

    if digit == 0
        prior=prior1;
        transmat=transmat1;
        means=mu1;
        Sigma=Sigma1;
        mixmat=mixmat1;
    end

    prior(:,:,digit+1)=prior1;
    transmat(:,:,digit+1)=transmat1;
    means(:,:,:,digit+1)=mu1;
    Sigma(:,:,:,:,digit+1)=Sigma1;
    mixmat(:,:,digit+1)=mixmat1;


end

%Se cargan los digitos de reconocimiento
listfiles=dir (['FeatPrueba\*.mat']);

for i=1:length(listfiles)
     load(['FeatPrueba\' listfiles(i).name]);
     X=X';
     mrep=repmat(m,[1 size(X,2)]);
     srep=repmat(s,[1 size(X,2)]);

%Se normalizan los vectores
     X=(X-mrep)./srep;
```

```
%Se calculan las probabilidades de que la muestra haya sido...
%generada por el HMM correspondiente al digito "digit"

  for digit=1:10
     loglik(digit) = mhmm_logprob(X, prior(:,:,digit), transmat(:,:,digit),...
      means(:,:,:,digit), Sigma(:,:,:,:,digit), mixmat(:,:,digit));
  end
 %Se localiza el modelo que mejor explica el dato
  [c,num]= max(loglik);

  %Se decodifica para dar el digito reconocido
  disp(num2str(num-1));

end
```

Apéndice G

Programa de reconcimiento del corpora E con HMM

```
clear all;
clc;

Wall=[];
%Se calcula la media y desviacion estandar de todas las muestras
for digit=0:9
   load(['Feat\' num2str(digit) '.mat']);
   %W=X;
   Wall=[Wall W];
end
  m=mean(W,2);
  s=std(W,0,2);
  clear Wall W X;

O=37;
Q=4;
M=1;
left_right=0;

for digit=0:9

 load(['Longitudes\' num2str(digit) 'long.mat']);
 load(['Feat\' num2str(digit) '.mat']);
```

```
 T=size(W,2);
 mrep=repmat(m,[1 T]);
 srep=repmat(s,[1 T]);

 %Se normalizan los vectores de caracteristicas en R^O
 Wtilde=W;
 Wtilde=(W-mrep)./srep;

%Los datos de entrenamiento se guardan en arreglos de celdas
    for i=1:73
         datacell{1,1,i}=Wtilde(:,longitudes(1,i)+1:longitudes(1,i+1));
    end

%Se inicializa el vector de probabilidad incial a (1,0,...,0)
    prior0 = eye(Q,1);
%Modelo de Bakis
    transmat0 = mk_leftright_transmat(Q, rand); % funcion del HMMToolbox


%Se estima un solo modelo de Markov que de log-verosimilitud finita

      flag = 0;
    while flag == 0
    disp(['still in digit                ' num2str(digit)]);

    obj = gmdistribution.fit(Wtilde',Q*M,'CovType','diagonal',...
    'Regularize',.01);
    mu0=obj.mu;
    mu0=mu0';
    Sigma0=obj.Sigma;

    SigmaM=diag(Sigma0(:,:,1));
    for i=2:Q*M
        SigmaM(:,:,i)=diag(Sigma0(:,:,i));
    end;
    Sigma0=SigmaM;
```

```
    mu0 = reshape(mu0, [O Q M]);
    Sigma0 = reshape(Sigma0, [O O Q M]);
    mixmat0 = mk_stochastic(rand(Q,M)); % funcion del HMMToolbox

    [LL, prior1, transmat1, mu1, Sigma1, mixmat1] = ...
    mhmm_em(datacell, prior0, transmat0, mu0,Sigma0, mixmat0,...
    'adj_prior',0, 'verbose',0,'max_iter', 10); % funcion del HMMToolbox
    if LL(end)~= -Inf
       flag =1;
    end

    end

    if digit == 0
        prior=prior1;
        transmat=transmat1;
        means=mu1;
        Sigma=Sigma1;
        mixmat=mixmat1;
    end

    prior(:,:,digit+1)=prior1;
    transmat(:,:,digit+1)=transmat1;
    means(:,:,:,digit+1)=mu1;
    Sigma(:,:,:,:,digit+1)=Sigma1;
    mixmat(:,:,digit+1)=mixmat1;
 %Se guardan los parametros de los HMM en multi-arrays

end

%Se cargan los digitos de reconocimiento
listfiles=dir (['FeatTest\*.mat']);

for i=1:length(listfiles)
     load(['FeatTest\' listfiles(i).name]);
     X=X';
     mrep=repmat(m,[1 size(X,2)]);
     srep=repmat(s,[1 size(X,2)]);
```

```
 %Se normalizan los vectores
     X=(X-mrep)./srep;

%Se calculan las probabilidades de que la muestra haya sido
%generada por el HMM correspondiente al digito "digit"
     for digit=1:10
     loglik(digit) = mhmm_logprob(X, prior(:,:,digit),...
      transmat(:,:,digit), means(:,:,:,digit),...
       Sigma(:,:,:,:,digit), mixmat(:,:,digit)); % funcion del HMMToolbox
  end
%Se localiza el modelo que mejor explica el dato
  [c,num]= max(loglik);
%Se decodifica para dar el digito reconocido
  disp(num2str(num-1));

end
```