Cadenas Ocultas de Markov aplicadas al reconocimiento de voz

Mauricio Ramos C.I. 3000693

Cadenas Ocultas de Markov aplicadas al reconocimiento de voz

Resumen

El presente trabajo es una introducción a las técnicas de reconocimiento de voz, que se encuentra orientado específicamente al reconocimiento de dígitos aislados. Dicho reconocimiento se basa en comparar dos emisiones vocales aisladas a fin de determinar si corresponden o no a la misma palabra. Con el fin de efectuar esta comparación se crea un sistema de reconocimiento de voz basado en cadenas ocultas de Markov HMM, por sus siglas en inglés, y la utilización de características MFCC , Mel Frequency Cepstral Coefficients, para la manipulación de las señales.

Introducción

Recientemente, como resultado de la amplia variedad de ordenadores, se descubren diversas formas de intercambio de información entre el hombre y el ordenador. Con lo cual, el insertar datos en el ordenador a través del habla y su posterior reconocimiento por el equipo se ha vuelto uno de los campos científicos en desarrollo. En la actualidad una gran variedad de sistemas de reconocimiento de voz son utilizados en aplicaciones telefónicas: agencias de viajes, atención al cliente, información etc. La mejoría de estos sistemas de reconocimiento del habla han ido aumentando y su eficacia cada vez es mayor. Cabe destacar también que el reconocimiento automático de voz involucra a varias disciplinas tales como psicología, acústica, procesamiento de señales, reconocimiento de patrones y lingüística. Por lo que la dificultades provienen de varios aspectos de estas áreas.

A continuación se describen, teniendo en cuenta diferentes aspectos, algunas características de los sistemas de reconocimiento automático de voz:

1. La forma en que el usuario le habla a la máquina. Existen básicamente tres formas:
    - Palabra Aislada: el usuario habla palabras individuales (o frases) tomadas de un vocabulario determinado.
    - Palabras Conectadas: el usuario habla en forma fluida una sucesión de palabras pertenecientes a un vocabulario restringido (e.g. dígitos telefónicos).
    - Habla continua: el usuario habla fluidamente usando palabras de un vocabulario grande (usualmente ilimitado).
2. Tamaño del vocabulario de reconocimiento
    - Pequeño: capaz de reconocer hasta 100 palabras.
    - Mediano: entre 100 y 1000 palabras.
    - Grande: más de 1000 palabras.
3. El conocimiento de los patrones de voz del usuario
    - Sistemas dependientes del locutor: adaptados a locutores particulares.
    - Sistemas independientes de locutor: trabajan con un población de locutores grande, la mayoría de los cuales son desconocidos para el sistema.
    - Sistemas adaptables: se adaptan al locutor particular mientras el sistema está en uso.
4. Grado de conocimiento acústico-lingüístico usado por el sistema.
    - Sólo conocimiento acústico. No usan conocimiento lingüístico.
    - Integración de conocimiento acústico y lingüístico. El conocimiento lingüístico está usualmente representado por restricciones sintácticas y semánticas sobre la salida del sistema de reconocimiento.
5. Grado de diálogo entre el usuario y la máquina.
    - Unidireccional (o pasivo). El usuario habla y la máquina realiza una acción como respuesta.

- Sistema de diálogo activado por la máquina. El sistema es el iniciador del diálogo, requiriendo información del usuario vía una entrada verbal.
- Sistema de diálogo natural. La máquina “conversa” con el locutor, le solicita entradas, actúa en función de las entradas y trata de clarificar ambigüedades.

Cabe destacar que cuando se plantea el reconocimiento automático de voz se está frente a una tarea inherentemente difícil, donde se plantean varios problemas debido principalmente a la variabilidad de las señales de voz. Algunas fuentes de variabilidad incluyen:

- Variabilidad en un locutor en mantener una pronunciación consistente y en el uso de palabras y frases.
- Variabilidad entre locutores debido a diferencias fisiológicos (e.g. diferente longitud del tracto vocal), acentos regionales, idiomas extranjeros, etc.
- Variabilidad entre transductores cuando se habla frente a diferentes micrófonos o aparatos telefónicos.
- Variabilidad introducida por el sistema de transmisión (redes de comunicación teléfonos celulares, etc.).
- Variabilidad en el ambiente, que incluyen conversaciones extrañas y eventos acústicos de fondo, como ruidos, etc.

Tradicionalmente se utilizan tres enfoques para abordar el reconocimiento automático de voz:

1. Enfoque Acústico-Fonético
2. Enfoque de Reconocimiento de patrones
3. Enfoque de inteligencia artificial

En este trabajo se utiliza el enfoque de reconocimiento de patrones, el cual consiste básicamente en dos pasos:

- Primer Paso: entrenamiento de patrones
- Segundo Paso: comparación de patrones

La característica principal de este enfoque es que usa un marco matemático bien definido y que establece representaciones consistentes de los patrones de voz que pueden usarse para comparaciones confiables a partir de un conjunto de muestras rotuladas, usando algoritmos de entrenamiento. La representación de los patrones de voz puede ser una plantilla (template), o un modelo estadístico (HMM: Hidden Markov Model), que puede aplicarse a un sonido (más pequeño que una palabra), una palabra, o una frase. En la etapa de comparación de patrones se realiza una comparación directa entre la señal de voz desconocida (a reconocer) y todos los posibles patrones aprendidos en la etapa de entrenamiento, de manera de determinar el mejor ajuste de acuerdo a algún criterio.

Descripción del problema

Se cuenta con un conjunto finito de 220 audios con señales de voz provenientes de 10 locutores. Estas señales consisten en palabras simples provenientes de un vocabulario de 11 palabras. El objetivo de este trabajo es la construcción de un sistema de reconocimiento de voz que permita identificar dichas palabras aisladas, introducir un análisis de dicho sistema y la muestra de los resultados obtenidos.

Alcance del trabajo

Se pretende construir un sistema de reconocimiento de voz de palabras aisladas, pequeño, independiente del locutor, solo con conocimiento acústico y unidireccional con un enfoque de reconocimiento de patrones basado en HMM. Que permita reconocer dígitos aislados del idioma inglés.

Metodología

Datos

Se cuenta con un conjunto finito de 220 audios con señales de voz provenientes de 10 locutores. Estas señales consisten en palabras simples provenientes de un vocabulario de 11 palabras, claramente expresadas y de duración bien definida, los digitos: “zero” a “nine” con el agregado de una señal de audio “oh”. Cada palabra es repetida dos veces por 5 mujeres y 5 hombres.

Se toman 3 mujeres y 3 hombres para generar un conjunto de entrenamiento de 132 instancias, 12 instancias para cada dígito. Dejando 2 mujeres y 2 hombres para el conjunto de test, 88 instancias en total, 8 por dígito.

Fundamento Teórico

Los Modelos Ocultos de Markov (Hidden Markov Models - HMM) son de gran utilidad en aplicaciones de reconocimiento automático de voz. Estos modelos consisten en dos procesos estocásticos anidados, donde cada vector característico (observación) es a su vez una función estocástica de cada estado del modelo. La función estocástica subyacente no es directamente observable, es decir, está oculta y sólo puede observarse a través de otro conjunto de procesos estocásticos que producen la secuencia de observación $\boldsymbol{o_t}$ en el instante t. Puede pensarse a un modelo HMM como una máquina de estados finitos en la cual cada estado en el caso más general, tiene asociada una Función de Densidad de Probabilidad (FDP) para cada vector característico.

Elementos de un HMM

1. N, es el número de estados en el modelo
2. q_t, al etiquetar cada estado como $\{1, 2, ..., N\}$, se denota al estado en el tiempo t como q_t
3. M, número posible de observación de símbolos distintos por estado, o igualmente, el tamaño del alfabeto discreto. Se denota individualmente a los símbolos como $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_M}\}$.
4. $A = \{a_{ij}\}$, matriz con la distribución de probabilidad de transición de un estado a otro, donde $a_{ij} = P[q_{t+1} = j \mid q_t = i]$, $1 \le i, j \le N$
5. $B = \{b_j(k)\}$, matriz con la distribución de probabilidad de observación de los símbolos, donde $b_j(k) = P[\boldsymbol{o_t} = \boldsymbol{v_k} \mid q_t = j]$, $1 \le k \le M$ define la pdf del símbolo $\boldsymbol{v_k}$ en el estado j.
6. $\pi = \{\pi_i\}$, distribución inicial de probabilidades para cada estado, $\pi_i = P\{q_1 = i\}$, $1 \le i \le N$
7. $\lambda = (A, B, \pi)$, parámetros de HMM, notación compacta de las tres medidas de probabilidad A, B y π
8. $\boldsymbol{O} = (\boldsymbol{o_1 o_2 ... o_T})$, secuencia observada, donde cada observación $\boldsymbol{o_t}$ es un símbolo de V, y T es el número de observaciones en la secuencia.
9. $\alpha_t(i)$, es la probabilidad de estar en el estado i con la observación $\boldsymbol{o_1 o_2 ... o_t}$ en el tiempo t dado el modelo λ, o sea: $\alpha_t(i) = P(\boldsymbol{o_1 o_2 .... o_t}, q_t = i | \lambda)$

10. $\beta_t(i)$, es la probabilidad de obtener en el resto de la observación $\boldsymbol{o_{t+1}o_{t+2}...o_T}$ dado que se está en el estado i en el tiempo t dado el modelo λ , o sea:
$\beta_t(i)=P(\boldsymbol{o_{t+1}o_{t+2}...o_T}|q_t=i,\lambda)$

Los tres problemas básicos de HMM

1. Dada la secuencia $\boldsymbol{O}=(\boldsymbol{o_1o_2...o_T})$ y el modelo $\lambda=(A,B,\pi)$, ¿como se debe computar eficientemente $P(\boldsymbol{O}|\lambda)$, la probabilidad de esa observación dado el modelo?
2. Dada la secuencia $\boldsymbol{O}=(\boldsymbol{o_1o_2...o_T})$ y el modelo $\lambda=(A,B,\pi)$, ¿como se debe elegir la correspondiente secuencia de estados $\boldsymbol{q}=(q_1q_2....q_T)$ de tal manera que optimice algún criterio (que mejor "explique" las observaciones) ?
3. ¿Como se deben ajustar los parámetros del modelo $\lambda=(A,B,\pi)$ para maximizar $P(\boldsymbol{O}|\lambda)$?

Solución al problema 1

Considerando una secuencia de estados fija

$$\boldsymbol{q}=(q_1q_2....q_T)$$

La probabilidad de la secuencia observada $\boldsymbol{O}=(\boldsymbol{o_1o_2...o_T})$ dada la secuencia de estados anterior, asumiendo independencia entre las observaciones es,

$$P(\boldsymbol{O}|\boldsymbol{q},\lambda)=\prod_{t=1}^{T}P(\boldsymbol{o_t}|q_t,\lambda)$$

$$P(\boldsymbol{O}|\boldsymbol{q},\lambda)=b_{q_1}(\boldsymbol{o_1}).b_{q_2}(\boldsymbol{o_2})....b_{q_T}(\boldsymbol{o_T})$$

La probabilidad de la secuencia de estados puede ser escrita como

$$P(\boldsymbol{q}|\lambda)=\pi_{q_1}a_{q_1q_2}.a_{q_2q_3}....a_{q_{T-1}q_T}$$

Con lo cual,

$$P(\boldsymbol{O},\boldsymbol{q}|\lambda)=P(\boldsymbol{O}|\boldsymbol{q},\lambda)P(\boldsymbol{q}|\lambda)$$

La probabilidad de $\boldsymbol{O}$ (dado el modelo) se obtiene sumando la ecuación anterior sobre todas las secuencias posibles $\boldsymbol{q}=(q_1q_2....q_T)$,

$$\boxed{P(\boldsymbol{O}|\lambda)=\sum_{all\,\boldsymbol{q}}P(\boldsymbol{O}|\boldsymbol{q},\lambda)P(\boldsymbol{q}|\lambda)}$$

Cabe observar que calcular directamente de la definición es computacionalmente costoso incluso para poco estados. De todas maneras existen procedimientos de cálculo computacional eficientes:

- "the forward procedure" que utiliza los $\alpha_t(i)$
- "the backward procedure" el cual utiliza las $\beta_t(i)$

Solución al problema 2

En principio podemos elegir entre varios criterios para optimizar la secuencia de estados $\boldsymbol{q}=(q_1q_2....q_T)$, una posible es por ejemplo el maximizar $P(\boldsymbol{q}|\boldsymbol{O},\lambda)$, lo cual es equivalente a maximizar $P(\boldsymbol{q},\boldsymbol{O}|\lambda)$. Esto se realiza con ayuda del algoritmo de Viterbi, definamos la cantidad

$$\delta_t(i)=\max_{q_1,q_2,...,q_{t-1}}P[q_1q_2...q_{t-1}\ ,\ q_t=i\ ,\ \boldsymbol{o_1o_2....o_t}\,|\,\lambda]$$

$\delta_t(i)$ indica dado que se está en el estado q_t cual es la "mejor" secuencia de estados previa, en términos de probabilidad. Por inducción se tiene

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

El procedimiento completo se muestra a continuación:

1. Inicialización
   $\delta_t(i)=\pi_i b_i(\boldsymbol{o_1}),\ \ 1\le i\le N$
   $\psi_1(i)=0$, esta variable nos ayuda a recuperar la secuencia de estados
2. Recursividad
   $\delta_t(j)=\max_{1\le i\le N}\ [\delta_{t-1}(i)a_{ij}].b_j(\boldsymbol{o_t}),\ \ \begin{matrix}2\le t\le T\\ 1\le j\le N\end{matrix}$
   $\psi_t(j)=arg\ \max_{1\le i\le N}\ [\delta_{t-1}(i)a_{ij}],\ \ \begin{matrix}2\le t\le T\\ 1\le j\le N\end{matrix}$, devuelve el estado q_i que maximizo $[\delta_{t-1}(i)a_{ij}]$
3. Terminación
   $P^{opt}=\max_{1\le i\le N}\ [\delta_T(i)]$
   $q_T^{opt}=arg\ \max_{1\le i\le N}\ [\delta_T(i)]$
4. Camino de secuencias anteriores
   $q_t^{opt}=\psi_{t+1}(q_{t+1}^{opt}),\ \ t=T-1,\ T-2,\ \ldots,\ 1$

Solución al problema 3

El tercer problema de HMMs es el que ofrece más dificultades, en el mismo se necesita determinar un método para ajustar los parámetros del modelo $\lambda=(A,B,\pi)$ de manera que satisfagan ciertos criterios de optimización. En otras palabras se intenta optimizar los parámetros del modelo para describir de mejor forma como se construye una secuencia de observación dada. La secuencia de observación usada para ajustar los parámetros del modelo es llamada la secuencia de entrenamiento porque es usada para entrenar el HMM. El problema del entrenamiento es una de las aplicaciones más cruciales para el HMM, porque nos permite adaptar de manera óptima los parámetros del modelo que son observados durante el entrenamiento de datos. Cabe observar que no se conoce una manera cerrada de resolver analíticamente la maximización de $P(\boldsymbol{O}|\lambda)$ (probabilidad de la secuencia de observaciones). Por lo que se utilizara el método de máxima verosimilitud para encontrar algún juego de parámetros $\lambda=(A,B,\pi)$ que produzcan un máximo local de $P(\boldsymbol{O}|\lambda)$.

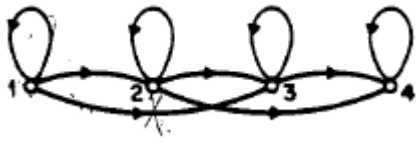
Un manera razonable para re-estimar π , A y B puede ser la siguiente:

- $\bar{\pi}_i=nro\ de\ veces\ en\ el\ estado\ \boldsymbol{i}\ en\ el\ tiempo\ \boldsymbol{t}=\boldsymbol{1}$
- $\bar{a}_{ij}=\dfrac{nro\ esperado\ de\ transiciones\ del\ estado\ \boldsymbol{i}\ al\ \boldsymbol{j}}{nro\ esperado\ de\ transiciones\ desde\ el\ estado\ \boldsymbol{i}}$
- $\bar{b}_j(k)=\dfrac{nro\ esperado\ de\ veces\ en\ el\ estado\ \boldsymbol{j}\ observando\ el\ símbolo\ \boldsymbol{v_k}}{nro\ esperado\ de\ veces\ en\ el\ estado\ \boldsymbol{j}}$

Donde se utiliza el modelo actual $\lambda=(A,B,\pi)$ para calcular el lado derecho de las ecuaciones anteriores, obteniendo así el nuevo modelo $\bar{\lambda}=(\bar{A},\bar{B},\bar{\pi})$. Basados en este procedimiento se puede iterar usando en cada paso $\bar{\lambda}$ en lugar de , λ al repetir los cálculos de re-estimación. De esta manera se puede aumentar $P(\boldsymbol{O}|\lambda)$ hasta encontrar un máximo local.

Modelo de Bakis

A lo largo de este trabajo se estructura la HMM a través del modelo de Bakis, pues este modelo tiene características deseables para modelar fácilmente señales cuyas propiedades cambian para todo tiempo de manera sucesiva, es decir, para un incremento temporal el índice de estados se incrementa o se mantiene sin cambio. Naturalmente esto ocurre en señales como el habla.



Bakis model

Las características del modelo de Bakis imponen limitaciones sobre $A=\{a_{ij}\}$ y sobre $\pi=\{\pi_i\}$:

- $a_{ij}=0\,,\quad j<i$, esto implica que la diagonal inferior de A es nula
- $\pi_1=1 \;\; con \;\; \pi_i=0, \quad i\neq 1$, dado que la secuencia de estados comienza en 1 (y termina en N)

A menudo se suelen imponer restricciones adicionales en las probabilidades de transición de estados para asegurar que no se producen cambios grandes en los índices en las transiciones. Por ejemplo se suele imponer una restricción de la forma $a_{ij}=0 \quad j>i+\Delta$

En particular, para el caso de figura anterior es Δ = 2, es decir, no se permiten saltos de más de dos estados. Resulta claro que para el último estado en el modelo izquierda-a- derecha se verifica

$a_{NN}=1 \quad , \quad a_{Nk}=0, \;\; k<N$

Sin embargo la mayor limitante es que **no se puede usar una única observación para entrenar el modelo**. Esto se debe a que la naturaleza transitoria de los estados dentro del modelo permite solo un pequeño número de observaciones en cualquier estado. Por lo que para tener estimaciones fiables de todos los parámetros se deben utilizar varias secuencias de observación.

Utilización de varias secuencias de observación

Se denota un set de K secuencias de observaciones como $\boldsymbol{O}=[\boldsymbol{O^1},\boldsymbol{O^2},\ldots,\boldsymbol{O^K}]$ donde $\boldsymbol{O^k}=\left(\boldsymbol{o_1^k}\,\boldsymbol{o_2^k}\ldots\boldsymbol{o_{T_k}^k}\right)$ es la k-ésima secuencia de observación, también se asume que las secuencias de observación son independientes entre si.

Bajo estas consideraciones se puede demostrar que las fórmulas para estimar los parámetros son las siguientes:

- $\bar{\pi}_i$ no es iterada pues dado el modelo de Bakis $\boxed{\pi_1=1 \;\; con \;\; \pi_i=0, \quad i\neq 1}$
- $$\boxed{\bar{a}_{ij}=\frac{\sum_{k=1}^{K}\left[\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\,a_{ij}\,b_j(\boldsymbol{o_{t+1}^k})\beta_{t+1}^k(j)\right]}{\sum_{k=1}^{K}\left[\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\beta_t^k(i)\right]}}$$

- $$\bar{b}_j(l)=\frac{\sum_{k=1}^{K}\left[\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\beta_t^k(i)\delta(\boldsymbol{o_t},\boldsymbol{v_k})\right]}{\sum_{k=1}^{K}\left[\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\beta_t^k(i)\right]}$$ donde $$\delta(\boldsymbol{o_t},\boldsymbol{v_k})=\begin{cases}1, & si\ \boldsymbol{o_t}=\boldsymbol{v_k}\\ 0, & si\ \boldsymbol{o_t}\neq\boldsymbol{v_k}\end{cases}$$

Densidades de probabilidad de observación continuas en HMMs

En lo desarrollado hasta el presente se asumió que las observaciones estaban caracterizadas por símbolos discretos dentro de un alfabeto finito, por lo que se podían usar densidades de probabilidad discretas dentro de cada estado del modelo. El problema con este enfoque es que para la mayoría de las aplicaciones (y en particular para reconocimiento de palabra) las observaciones son señales (vectores) continuos. Si bien se pueden cuantizar usando un libro de códigos (Vector Quantization) esto trae aparejada una seria degradación de la Quantization performance de los dispositivos de reconocimiento. Un enfoque muy usado, es representar la función de densidad de probabilidad continua como una combinación lineal de distribuciones Gaussianas (mezclas Gaussianas) de la forma:

$$b_j(O)=\sum_{m=1}^{M}c_{jm}\boldsymbol{N}(O,\mu_{jm},U_{jm});\quad 1\le j\le N$$

Donde O es el vector de observación, c_{jm} , $m=1$, ..., M son los coeficientes de la mezcla de M Gaussianas en el estado j , y $\boldsymbol{N}(O,\mu_{jm},U_{jm})$ es una distribución Gaussiana con media μ_{jm} y matriz de covarianza U_{jm} correspondiente a la m-ésima componente de la mezcla en el estado j . Los coeficientes c_{jm} deben satisfacer la restricción estocástica:

$$\sum_{m=1}^{M}c_{jm}=1;\quad 1\le j\le N$$

$$c_{jm}\ge 0,\quad 1\le j\le N\ ,\ 1\le m\le M$$

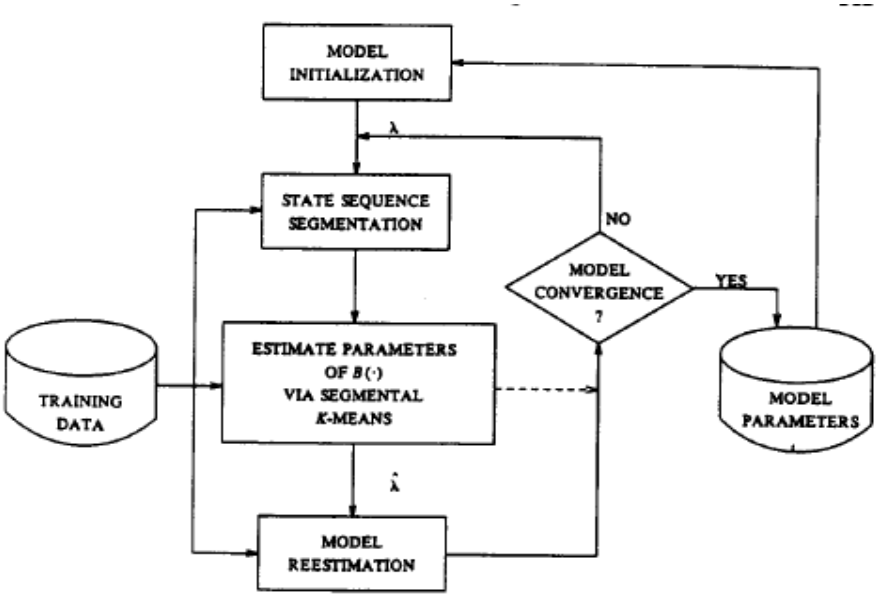
de manera que resulte

$$\int_{-\infty}^{\infty}b_j(x)\,dx=1,\quad 1\le j\le N$$

Esta representación puede ser usada para aproximar, con una precisión arbitraria, cualquier función de densidad de probabilidad continua finita. Se han desarrollado fórmulas de re-estimación para los parámetros μ_{jm} , U_{jm} y c_{jm} de las mezclas Gaussianas. En la siguiente sección se muestra como se calculan.

Segmental K-Means segmentación en estados.

Una buena inicialización de los parámetros de las densidades $b_j(\boldsymbol{o}_t)$ son esenciales para una rápida y apropiada convergencia en las fórmulas de reestimación. Un procedimiento que provee buenas estimaciones iniciales de estos parámetros se muestra en la siguiente figura,



Procedimiento de entrenamiento por segmentación K-Means utilizado para estimar los valores de los parámetros para un ajuste óptimo de la mezcla de densidades continuas a un número finito de secuencias de observaciones.

El procedimiento de entrenamiento es una variante del conocido procedimiento iterativo K-Means para clustering data.

Se asume que se tiene un conjunto de observaciones de entrenamiento, y una estimación inicial de todos los parámetros del modelo. Sin embargo a diferencia de lo que se requiere para re-estimación, las estimaciones iniciales del modelo pueden ser elegidas en forma aleatoria o en base a cualquier modelo apropiado para los datos.

El resultado de segmentar cada secuencia de entrenamiento es, para cada uno de los N estados, una estimación por máxima verosimilitud del conjunto de observaciones que ocurren dentro de cada estado j conforme el modelo actual. Cuando se usa observaciones basadas en densidades continuas, el procedimiento de segmentación por K-Means es usado para agrupar los vectores de observación dentro de cada estado j en un conjunto de M clusters (usando una medida Euclideana deformada), donde cada cluster representa una de las M mezclas de las densidades de $b_j(\boldsymbol{o}_t)$. Desde el clustering, se deriva la actualización de los parámetros:

$$\hat{c}_{jm} = \frac{\textit{número de vectores clasificados en el cluster m del estado } j}{\textit{número de vectores en el estado } j}$$

$$\hat{\mu}_{jm} = \textit{promedio de las muestras de los vectores clasificados en el cluster m del estado } j$$

$$\hat{U}_{jm} = \textit{matriz de covarianza de las muestras de los vectores clasificados en el cluster m del estado } j$$

Basados en esta segmentación, las actualizaciones estimadas para los coeficientes a_{ij} pueden ser obtenidas contando el número de transiciones del estado i al estado j y dividiendo por el número de transiciones cuyo origen es el estado i .

Una actualización del modelo $\bar{\lambda}$ es obtenida desde los nuevos parámetros del problema. Si la distancia entre los modelos excede un cierto umbral, entonces el modelo viejo λ es remplazado por el nuevo $\bar{\lambda}$ y todo el loop de entrenamiento es repetido. Cuando en algún paso no se supera el umbral, se asume una convergencia del modelo y se tiene un modelo final.

Características MFCC en sistemas de reconocimiento de voz

La parte central de un sistema de reconocimiento de voz consiste en entrenar y reconocer procesos. El comportamiento de estos procesos y la exactitud del reconocimiento de voz, dependen significativamente de la elección de características a utilizar. Lo más natural podría ser el uso de la función temporal de la señal de voz como una características, pero esto no es efectivo, la razón es que una misma persona dice las mismas palabras, variando significativamente la función temporal. Actualmente el método de calcular MFCC (Mel Frequency Cepstral Coefficients) es ampliamente utilizado como característica del habla en reconocimiento de voz y es el que va a utilizarse en este trabajo.

Se puede modelar la generación del habla como consistente en dos partes: la generación de la señal de excitación y el filtrado de la misma en tracto vocal. Si $e(n)$ denota la señal de una secuencia de excitación y $\theta(n)$ denota la respuesta impulsiva del tracto vocal, se tiene una secuencia de voz,

$$s(n)=e(n)*\theta(n)$$

donde si pasamos a frecuencia,

$$S(\omega)=E(\omega).\Theta(\omega)$$

En el método MFCC se utiliza el logaritmo del la ecuación anterior, por lo que la multiplicación del espectro se vuelve aditiva,

$$\log|S(\omega)|=\log|E(\omega)|+|\Theta(\omega)|$$

Por lo que es posible separar el espectro de la excitación del espectro del tracto vocal

$E(\omega)$ es responsable de las variaciones rápidas del espectro mientras que $\Theta(\omega)$ de las lentas.

Luego de aplicar logaritmo y la transformada inversa de Fourier se tiene un nuevo dominio llamado cepstrum domain, y la palabra quefrency es usada para describir las “frecuencias” en el cepstrum domain.

Algoritmo para el cálculo de características

1. **Pre-énfasis**. El espectro de una señal de voz es predominantemente de baja frecuencia, por lo que utiliza un filtro FIR pasa alto, para nivelar el espectro.

   $$s_p(n) = s_{en}(n) - \alpha . s_{en}(n-1)$$

   donde α es el coeficiente del filtro ($\alpha \in (0{,}95\,;1)$) y $s_{en}(n)$ es la señal de entrada.

2. **Voice activation detection.** El problema de localizar el inicio o el fin de una palabra es un problema bastante importante. Una mala determinación de estas variables puede decrementar la performance del reconocedor de voz. Algunas medidas utilizadas para solventar este problema son la energía, la potencia y los cruces por cero, parciales. A continuación se muestran sus cálculos:

   $$E_s(m)=\sum_{m}^{n=m-L+1} s_p^2(n) \quad , \quad P_s(m)=\frac{1}{L}\sum_{m}^{n=m-L+1} s_p^2(n)$$

   $$Z_s(m)=\frac{1}{L}\sum_{m}^{n=m-L+1} \frac{sgn(s_p(n))-sgn(s_p(n-1))}{2}$$

donde

$$sgn(s_p(n)) = \begin{cases} 1, & si\ s_p(n) \geq 0 \\ 0, & si\ s_p(n) < 0. \end{cases}$$

Sobre cada bloque de L=100 muestras, estas medidas calculan algunos valores. Los cruces por cero tienden a aumentar durante regiones sin voz. La idea es generar un trigger para tomar alguna decisión con estas medidas, pero para ello se necesita información del ruido de fondo. Por lo que se asume que los 5 primeros bloques de la señal son ruido de fondo, con lo cual se calcula la media y la varianza del mismo. Se utiliza la siguiente descripción que combina varios detectores:

$$W_s(m) = P_s(m).(1 - Z_s(m)).S_c$$

Donde $S_c = 1000$ es un factor de escala. El trigger para esta función puede ser descripto como

$$t_w = u_w + \alpha \delta_w$$

u_w es la media y δ_w es la varianza de la función $W_s(m)$ calculada para los 5 primeros bloques, y α es un término constante que tiene la función de sintonizar acorde a las características de la señal.

$$\alpha = 0.2 \delta_w^{-0.4}$$

Luego Voice activation detection function $VAD(m)$, puede ser encontrada como

$$VAD(m) = \begin{cases} 1, & si\ W_s(m) \geq t_w \\ 0, & si\ W_s(m) < t_w \end{cases}$$

3. **Framing**. La señal de entrada se divide en frames superpuestos de N muestras.

$$s_{frame}(n) = s_p(n).w(n),$$

$$w(n) = \begin{cases} 1, & si\ K.r < n \leq K.r + N\ ,\ \ r = 0,\ 1,\ 2,\ldots,(numframe - 1) \\ 0, & en\ otro\ caso. \end{cases}$$

numframe es el número de frames, f_s es la frecuencia de muestreo, t_{frame} es el largo temporal de cada frame y K es el frame step. $N = f_s.t_{frame}$

Dado que nuestros datos están muestreados a $f_s = 8\text{kHz}$ se utilizará $N = 200$ y $K = 80$

4. **Enventanado.** Para suavizar las discontinuidades se utiliza una ventana de Hamming,

$$s_w(n) = \left\{ 0{,}54 - 0{,}46 \cos\left(\frac{2\pi(n-1)}{N-1} \right) \right\} s_{frame}(n),\ \ 1 \leq n \leq N$$

5. **Transformada rápida de Fourier (FFT).** Aplicando FFT a los frames enventanados se obtiene el espectro de los frames,

$$bin_k = \left| \sum_{n=1}^{N} s_w(n) e^{-j(n-1)k\frac{2\pi}{N}} \right|,\ \ k = 0,\ 1,\ 2,\ldots,(N-1)$$

6. **Filtrado de Mel.** Las componentes de baja frecuencia del espectro son ignoradas. Entre las frecuencias *minfreq* y *maxfreq* se dividen en forma equidistante NF canales sobre el dominio frecuencial de Mel. Estos canales se solapan a la mitad y sobre cada canal se aplica una ventana triangular. El centro de las frecuencias de los canales en términos de los índices de intervalos de FFT ($cbin_i$ para el canal i) se calcula a continuación.

$$Mel(x) = 2595 \log\left(1 + \frac{x}{700}\right)$$

$$f_{c_1} = Mel^{-1}\left\{ Mel\{f_{start}\} + \frac{Mel\{f_s/2\} - Mel\{f_{start}\}}{NF} i \right\}, \quad i = 1, 2, \ldots, (NF-1)$$

$$cbin_i = round\left\{ \frac{f_{c_1}}{f_s} N \right\} \quad \text{donde} \quad round(.) \quad \text{redondea al entero mas cercano.}$$

Donde la salida del filtro mel es,

$$fbank_k = \sum_{i=cbin_{k-1}}^{cbin_k} \frac{i - cbin_{k-1} + 1}{cbin_k - cbin_{k-1} + 1} bin_i + \sum_{i=cbin_k+1}^{cbin_{k+1}} \left(1 - \frac{i - cbin_k}{cbin_{k+1} - cbin_k + 1}\right) bin_i, \quad k = 1, 2, \ldots, (NF-1)$$

7. **Transformación no lineal.** Se aplica logaritmo neperiano a la salida del filtro mel

$$f_i = \ln(fbank_i), \quad i = 1, 2, \ldots, (NF-1)$$

8. **Coeficientes Cepstrales.** n cepstral coefficients son calculados a partir de DCT

$$C_i = \sum_{j=1}^{NF-1} f_j . \cos\left(\frac{\pi . i}{NF-1}(j - 0.5)\right), \quad i = 1, 2, \ldots n$$

9. **Cepstral Mean Subtraction (CMS).** Por último para eliminar el efecto del canal debido a diferentes tomas, se resta el promedio de los coeficientes mel-cepstrum,

$$mc_j(q) = C_j(q) - \frac{1}{M} \sum_{t=1}^{M} C_i(q), \quad q = 1, 2, \ldots n$$

Técnica

Para construir un sistema de reconocimiento de voz de palabras aisladas se utiliza un enfoque de reconocimiento de patrones, a través de un modelo estadístico utilizando HMM, extrayendo las características MFCC sobre muestras de las señales para poder compararlas.

Las líneas generales para obtener el reconocedor de palabras aisladas son las siguientes, se construirá una HMM λ_v - donde se debe estimar los parámetros del modelo $\lambda=(A,B,\pi)$ - con cada palabra v del vocabulario, optimizando la verosimilitud del vector de observaciones del conjunto de entrenamiento para la palabra v .

Cada palabra a reconocer, debe seguir el proceso que se muestra en la siguiente figura.

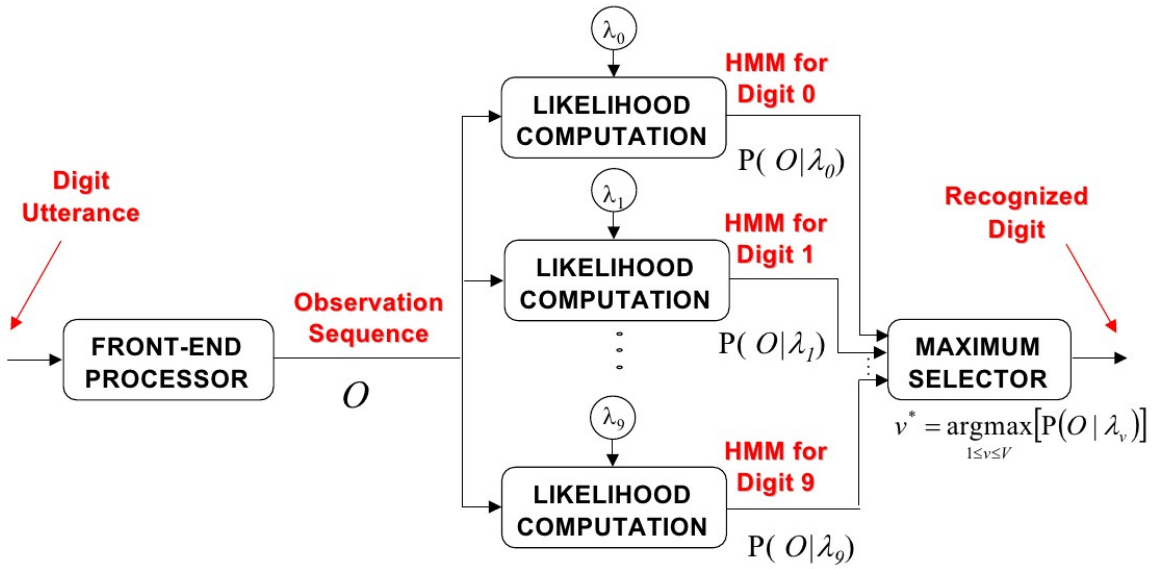


Diagrama de bloques de un reconocedor HMM de palabras aisladas

En principio para cada palabra de entrada se extrae de la señal de voz las características MFCC que conformarán la secuencia de observación $\boldsymbol{O}=(\boldsymbol{o}_1\boldsymbol{o}_2\ldots\boldsymbol{o}_T)$, siguiendo con el cálculo de verosimilitud de cada modelo $P(\boldsymbol{O}|\lambda_v)$, para luego quedarse con el modelo de máxima verosimilitud, esto es:

$$\tilde{v}=arg\ \max_{1\le v\le V}[P(\boldsymbol{O}|\lambda_v)]$$

Siendo $\tilde{v}$ la palabra seleccionada por el reconocedor a la salida del proceso.

En principio se cuenta con un vocabulario de $V=11$ palabras a reconocer,

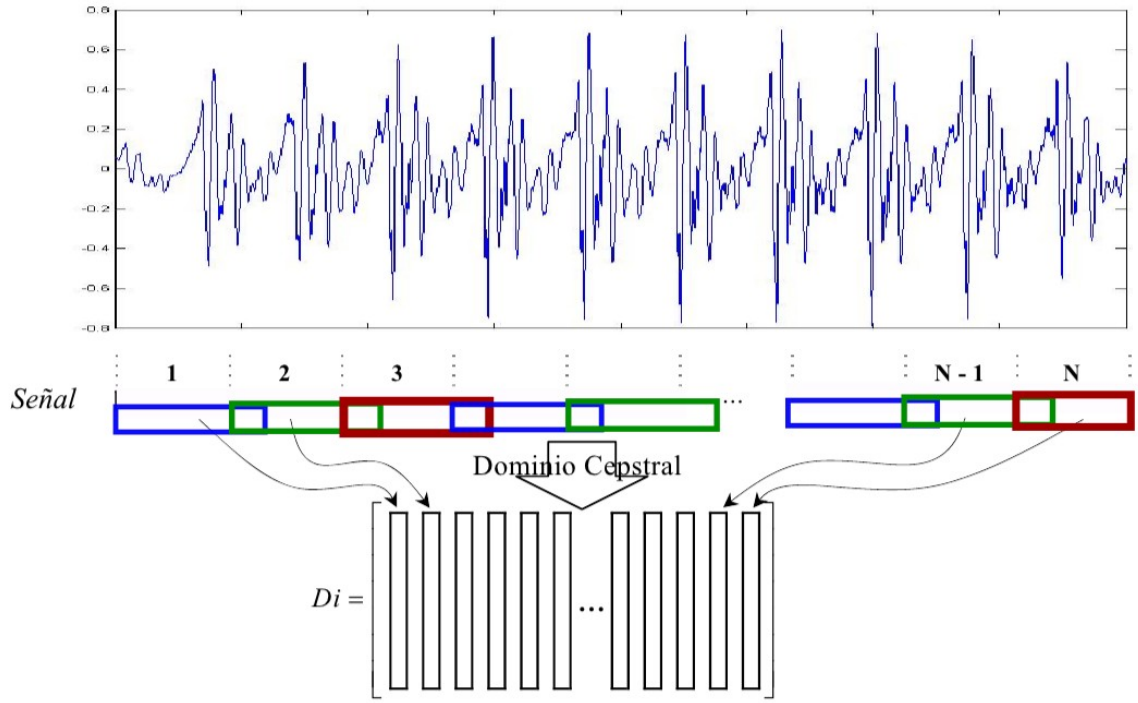
$V=\{'zero',\ 'one',\ 'two',\ 'three',\ 'four',\ 'five',\ 'six',\ 'seven',\ 'eight',\ 'nine',\ 'oh'\}$ donde para cada palabra se cuenta con un conjunto de $K=12$ instancias. Cada instancia constituye una secuencia de observación $\boldsymbol{O}=(\boldsymbol{o}_1\boldsymbol{o}_2\ldots\boldsymbol{o}_T)$ constituida por los coeficientes cepstrales MFCC de cada frame de la señal original de audio.

Obtención de MFCC

Para obtener estos coeficientes MFCC a partir de las señales de audio se utiliza básicamente el código de Dan Ellis “PLP and RASTA(and MFCC, and inversion) in Matlab”[1]. Excepto porque antes, se prepara la señal de cada instancia para quedarnos solo con la señal de voz. Para ello se utiliza un código con Voice activation detection, el cual no es implementado por el código RASTA. Se encuentra el principio y final de la señal de voz dentro de cada instancia y se permite un breve intervalo sin voz al inicio y al final del audio. Luego se actualizan todas las instancias con estas modificaciones. A partir de aquí se utiliza el código RASTA de Dan Ellis. Algunos de los parámetros ingresados son:

- el número de coeficientes cepstrales $numcep = 13$
- largo de frame $wintime = 0.025\text{s}$
- pasos entre frames sucesivos $hoptime = 0.010\text{s}$

El resto de los parámetros se dejan por defecto, dado que están bien distribuidos para nuestro tipo de señal. La siguiente figura muestra como se estructura a la salida del código RASTA, en la matriz D la información de los coeficientes cepstrales sobre cada instancia. Se considera N frames en la señal.



Estructura de la matriz D_i .

La i-ésima columna de la matriz D_i *contiene los coeficientes cepstral del i-ésimo frame*

1 http://labrosa.ee.columbia.edu/matlab/rastamat/

Construcción de HMM

Los sistemas de reconocimiento de voz basados en HMM constan principalmente de 3 etapas: Inicialización del modelo, Entrenamiento y Reconocimiento. Un toolbox que permite la implementación en Matlab de los HMM para reconocimiento de voz es el provisto por Kevin Murphy[2] , “Hidden Markov Model (HMM) Toolbox for Matlab”.

A continuación se describen brevemente las funciones del toolbox que permiten implementar las 3 etapas del reconocimiento de voz basado en HMM para una sola cadena.

INICIALIZACION : init_mhmm.m

Sintaxis:

[pi, A, B, mu, Sigma] = init_mhmm(data, N, M, cov_type)

Entradas:

- *N*: Número de estados del modelo
- *M*: Cantidad de mezclas Gaussianas en la distribución de probabilidad observación
- *cov_type*: Tipo de matriz de covarianza, puede ser de tipo diagonal (‘diag’), completa (‘full’) o esférica (‘spherical’).
- data: Datos de entrenamiento, deben ser ingresados como un arreglo de matrices [D1, D2, D3,..., Di , ...] donde cada matriz Di contiene los coeficientes cepstral de la señal de voz asociados a cada una de las observaciones de entrenamiento. Cada columna de la matriz Di contiene los coeficientes cepstral del frame respectivo (nro. filas = nro. coeficientes cepstral, nro. columnas = nro. de frames). Para ello se debe utilizar una variable tipo cell, data{i} = Di.

Salidas:

- *pi*: Distribución inicial de probabilidades de cada estado.
- *A*: Matriz de distribución de probabilidades de transición.
- *B*: Matriz de distribución de probabilidades de observación. Esta matriz contiene los c_{jm} , o sea los pesos de cada mixtura Gaussiana. Las filas son los estados j y las columnas el número de gaussiana m
- *mu*: Matriz con los valores medios asociados a las distribuciones Gaussianas. Las filas de *mu* son coef cepstrales, sus columnas son estados de la cadena y la tercer dimensión son las diferentes Gaussianas de la mixtura
- *sigma*: Matriz con los desvíos estándar asociados a las distribuciones Gaussianas. Las filas y columnas de sigma son coef cepstrales, la dimensión 3 son estados de la cadena y la dimensión 4 son las diferentes Gaussianas de la mixtura.

Cuando se utilizan los modelos HMM para el reconocimiento de señales de voz no todas las transiciones de estado son posibles, como ya se trato en el fundamento teórico lo recomendable es trabajar con la estructura de estados de Bakis para todas las cadenas HMM. Por tal motivo, es necesario restringir las estructuras de la matriz de transición $A=\left[a_{ij}\right]$ y de la matriz de probabilidad inicial $\pi=\left[\pi_i\right]$. Así, se incluyeron las siguientes líneas en el código de la función init_mhmm.m.

transmat = mk_leftright_transmat(Q,0.5);

2 http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html . El código presenta algunos errores, ver el apéndice para la corrección de los mismos.

init_state_prob = [1;zeros(Q-1,1)];

donde la primer línea inicializa la matriz de transición con las restricciones del modelo izquierda-a-derecha o modelo de Bakis y la segunda línea impone que la secuencia de estados comience en el estado 1.

init_mhmm presupone que los frames no estan etiquetados, por lo que aún no se sabe a que estado de la cadena pertenece cada frame. Una vez que se ingresa el número N de estados que posee la cadena, y el arreglo de matrices *data* con las características de todas las instancias de un mismo dígito, se juntan los frames de todas las instancias (estos deben tener el mismo número de coeficientes cepstrales), y luego se aplica K-Means. Esto genera los cluster utilizados para clasificar los frames. K = M*N es el número de clusters, el mismo depende de:

- M, cantidad de mezclas gaussianas que se utilizan en cada estado para representar las densidades de probabilidad de cada coeficiente cepstral.
- N número de estado de la cadena

Luego de ordenar la información se devuelve la estimación inicial de las variables que permiten construir las funciones de probabilidad de cada estado de la cadena a través de la mezcla de gaussianas.

- *mu* una matriz de dimensiones $[numcep \times N \times M]$
- *sigma* $[numcep \times numcep \times N \times M]$

Donde *numcep* es igual al número de coeficientes cepstrales utilizados. Este proceso permite un etiquetado automático de los frames además de caracterizar los estados de la cadena.

ENTRENAMIENTO: **mhmm_em.m**

Sintaxis

[ll_trace, pi, A, mu, sigma, B] = mhmm_em(data, pi0, A0, mu0, sigma0, B0,...)

Entradas:

- *data*: Datos de entrenamiento (ver explicación función init_mhmm.m).
- *pi0, A0, mu0, sigma0, B0*: son las matrices calculadas por la función de inicialización *init_mhmm.m.*

También pueden pasarse los siguientes parámetros opcionales:

- *'max_iter'* – número máximo de iteraciones [10]
- *'thresh'* – umbral de convergencia [1e-4]
- *'verbose'* – si es igual a 1, muestra en pantalla el valor de loglik en cada iteración [1]
- *'cov_type'* – Tipo de matriz de covarianza: 'full', 'diag' o 'spherical' ['full']

Por ejemplo, para que la cantidad máxima de iteraciones sea igual 30, la función debe ser llamada de la siguiente manera

mhmm_em(data, pi0, A0, mu0, sigma0, B0, 'max_iter', 30)

Esta función entrena la cadena utilizando el algoritmo Expectation-Maximization (EM), este método optimiza modelos con parámetros desconocidos iterando a partir de un estado inicial. En nuestro caso a partir de un cierto número de iteraciones se intenta encontrar un máximo local de la función de verosimilitud $P(\boldsymbol{O}|\lambda_v)$, al cambiar el modelo $\bar{\lambda}_v$, partiendo de las condiciones iniciales obtenidas con *init_mhmm.m* . El umbral de convergencia es el mínimo cambio de $\bar{\lambda}_v$ entre dos iteraciones consecutivas para que se considere que no se ha llegado a un máximo local.

<u>Reconocimiento:</u> mhmm_logprob.m

Sintaxis

[loglik, errors] = mhmm_logprob(data_rec, pi, A, mu, sigma, B)

Entradas:

- *data_rec*: Matriz con los coeficientes cepstral de la señal de voz a reconocer.
- *pi*, *A*, *mu*, *sigma*, *B*: Matrices que representan el modelo HMM entrenado.

Salidas:

- *loglik*: Verosimilitud logarítmica. Estima de la probabilidad de que la observación de entrada haya sido generada por el modelo.
- errors: Vector que indica en que casos la verosimilitud logarítmica haya sido infinito.

Una vez que se entreno la cadena HMM, se le pasa la instancia a reconocer a este código y nos devuelve la verosimilitud $P(\boldsymbol{O}|\lambda_v)$. Este código implementa las soluciones a dos de los problemas básicos en HMM expuestos en el fundamento teórico:

- el algoritmo de Viterbi, que encuentra la mejor secuencia de estados del modelo que explique la observación da la instancia a reconocer y
- el algoritmo Forward-Backward, el cual permite obtener un valor para la verosimilitud de la instancia de entrada dada la cadena HMM.

Enfoque

Los códigos mostrados en la sección anterior permiten construir los bloques que componen el reconocedor de palabras aisladas, y empezar a configurar el mismo para obtener los mejores resultados. Las variables que podemos modificar una vez armado el reconocedor son:

- N-El número de estados de cada cadena HMM
- M-El número de mezclas de Gaussianas para generar la función de distribución en cada estado de la cadena HMM

Cabe destacar que todas las cadenas tienen el mismo número de mezclas gaussianas.

Se hicieron las siguientes pruebas:

1. se toma el mismo número de estados (N) para todas las cadenas HMM asociadas a dígitos distintos, sin variar el número de mixturas Gaussianas (M).
2. se toma para cada cadena asociada a un dígito un número de estados N igual al número de fonemas, más dos estados asociados a silencios iniciales y finales del audio. Sin variar el número de mixturas Gaussianas (M).
3. Para el método que obtenga mejores resultados se prueba que sucede si se cambia el número de mixturas Gaussianas. O sea N fijo y M variable.
4. Para el método que de mejores resultados se prueba como afecta la performance el quitar la etapa VAD en la extracción de características.

Otra prueba que podría hacerse es no efectuar el “etiquetado” de los frames vía K-Means, sino etiquetar los frames de todas las instancias manualmente, de manera que se identifique a que fonema pertenecen. De esta manera seleccionar las condiciones iniciales del modelo HMM para cada dígito en función de estas etiquetas. Con lo cual garantizamos razonablemente que los estados queden caracterizados efectivamente por los fonemas. Si bien no por falta de tiempo no se explora dicha opción, se tiene claro que los códigos utilizados se mantendrían prácticamente sin cambios, exceptuando *init_mhmm.m*, donde se dejaría de clasificar los frames vía K-Means, y se tomaría la información de etiquetado de cada instancia para clasificar cada frame y obtener de esta manera una estimación inicial de las variables de la cadena a entrenar acorde a la distribución “empírica” de los fonemas. El código para entrenamiento se mantendría sin cambios al igual que el reconocimiento.

A continuación para correr toda las pruebas propuestas se introduce cada instancia de test 60 veces al reconocedor de voz y se promedia la verosimilitud a la salida. Cabe destacar que promediar la salida del entrenamiento sobre cada cadena no tiene sentido dado que disminuye la performance del reconocedor de voz. Esto último es coherente pues a la salida del entrenamiento se obtienen los parámetros del modelo que producen un máximo local de la función de verosimilitud $P(\boldsymbol{O}|\lambda_v)$, lo cual es deseable sobre la cadena de cada dígito . Es claro que al promediar máximos locales no tiene porqué obtenerse otro máximo local, por lo que si se implementase disminuiría la performance de cada cadena para identificar “su” dígito, empeorando en consecuencia la performance global del reconocedor.

Prueba 1

Las pruebas se efectúan al varia N en el intervalo de [1, 10][3], para poder describir el desempeño global del reconocedor se utilizan los datos de precisión y recall.

	Dígito a la entrada	
Dígito a la salida del sistema	TP, verdadero positivo	FP, falso positivo
reconocedor de voz	FN, falso negativo	TN, verdadero negativo

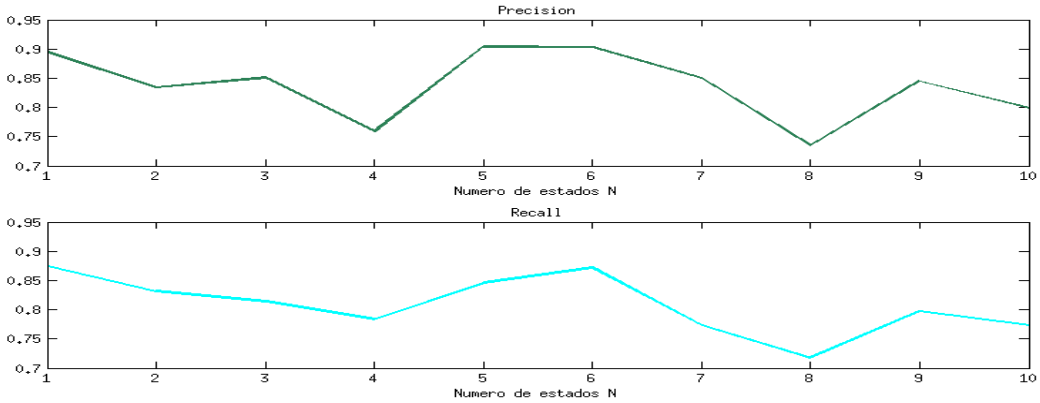
Donde

$$Presicion = \frac{TP}{TP + FP}$$

y

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

A continuación se muestra como varían estos parámetros al variar los estados de todas las cadenas HMM del sistema reconocedor de voz con N, manteniendo el número de mixturas gaussianas M=2.



3 En la sección 2 del apéndice se presentan todas las matrices de confusión que permitieron los cálculos

Estas gráficas nos muestran que si tomamos el mismo número de estados para cada cadena HMM asociada con algún dígito, conviene trabajar con N=5, otras opciones que producirían resultados similares son el estado N= 6 o 5, con un desempeño levemente menor del sistema reconocedor de voz.

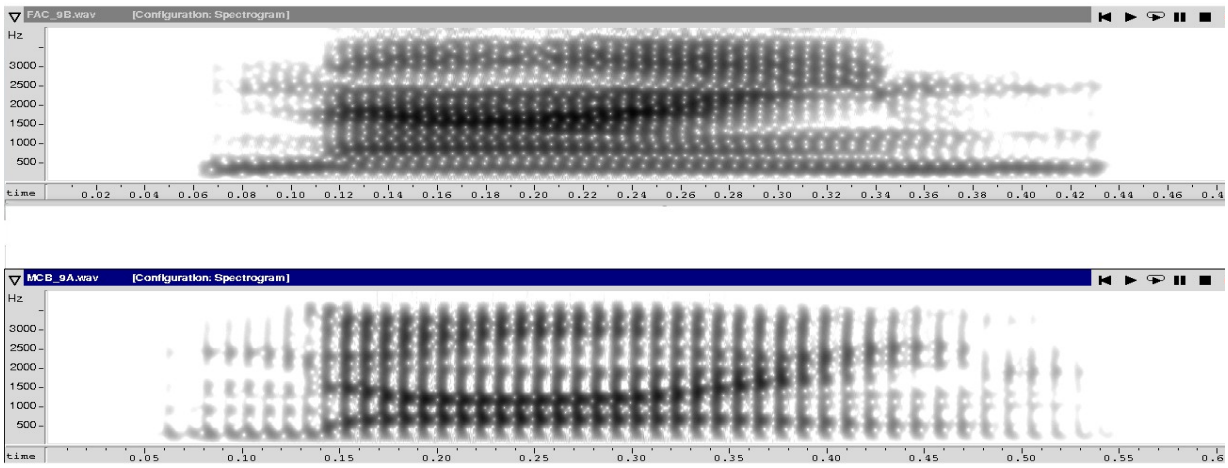
Número de estados (N)	Precisión	Recall
5	0.904	0.846
6	0.903	0.872
1	0.895	0.875

Es interesante observar que ocurre si tomamos una cadena de tan solo un estado, ¿que quiere decir tomar un solo estado por cadena?. Bien en principio al crear el sistema reconocedor de voz utilizando un único estado para todas las cadenas, básicamente se estaría comparando que tan verosímil es una entrada entre las distribuciones asociadas a cada dígito. No se necesitaría una HMM!. Probablemente este buen comportamiento para N=1 se deba al poco número de instancias por dígito en el conjunto de entrenamiento, produciendo poca variabilidad entre las instancias de un solo dígito.

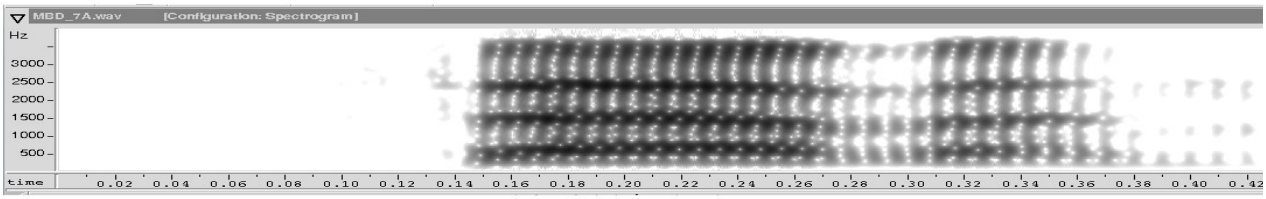
Cabe destacar que si bien el desempeño global es bastante bueno, surgen problemas individuales en el reconocimiento de dos dígitos, el dígito “two” y en peor manera el dígito “nine”

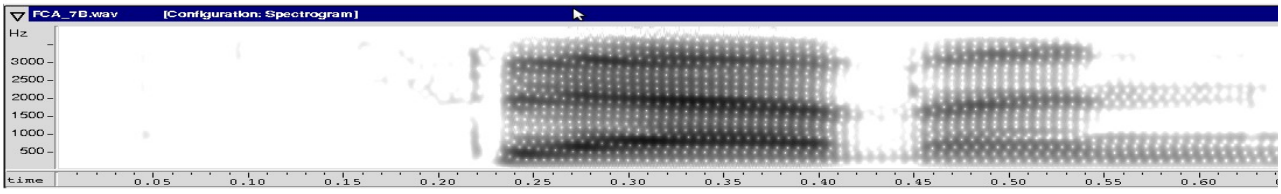
El sistema reconocedor tiende a confundir el dígito “two ” básicamente con el dígito “zero” y en menor medida con el dígito “seven”. Mientras que con el dígito “nine” el sistema reconocedor tiende a confundirlo con “seven”, “zero” y “one”.

A continuación se muestran los espectrogramas de algunas instancias de los dígitos “nine” , “seven”, “zero” y “one”.

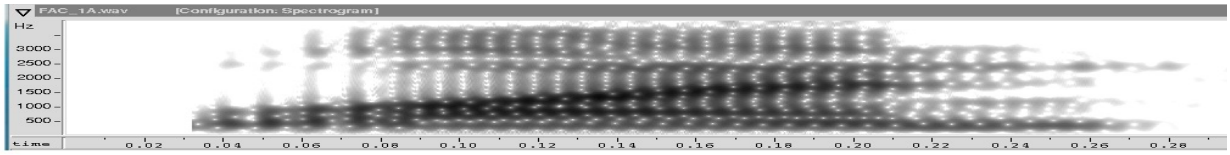


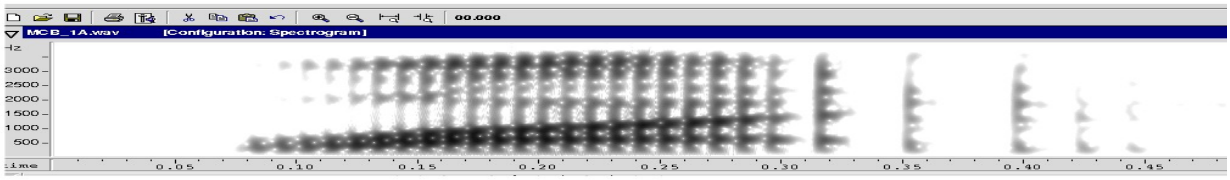
Espectrogramas del dígito "nine"



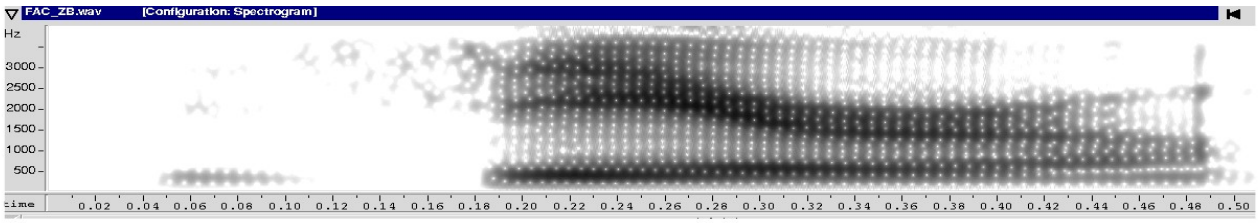


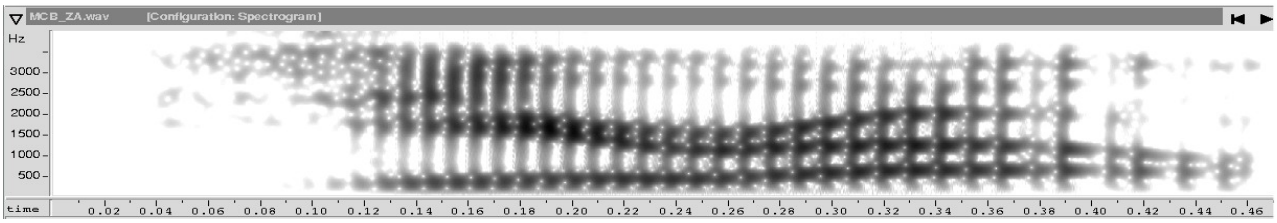
Espectrogramas del dígito "seven"





Espectrograma del dígito "one"





Espectrogramas del dígito "zero"

Inspeccionando los espectrogramas anteriores parece razonable pensar que “nine” comparte características con ciertas partes de los espectrogramas de “seven”, “zero” y “one”. Por lo que la mala clasificación del sistema de reconocimiento con el dígito “nine” probablemente se deba falta de entrenamiento con una cantidad mayor de instancias más representativas de este dígito.

Prueba 2

Ahora se crea un sistema de reconocimiento en el cual cada cadena HMM tiene un número de estados acorde al número de fonemas asociado al dígito que intenta reconocer. Se agregan dos estados extra para contemplar los posibles silencio inicial y final en las señales de audio de cada instancia. Los resultados a la salida del sistema [4] son los siguientes:

Número de estados (N)	Precisión	Recall
Acorde al número de fonemas	0.894	0.841

Se puede ver que el desempeño es levemente menor que el obtenido con un único estado por cadena HMM. Por lo que en principio no se ha mejorado la performance.
Con el fin de obtener un desempeño más alto se prueba lo siguiente, se exploran las matrices de confusión obtenidas en la sección anterior en busca de cambios, con el número de estados, en el desempeño individual de cada cadena, tomando el número de estado que maximice cada cadena. Luego se crea un sistema de reconocimiento donde cada cadena HMM tiene un número de estados que maximiza individualmente su funcionamiento.
En esta búsqueda se puede ver que básicamente, solo dos cadenas HMM no adquieren su máximo desempeño con el número de fonemas, el dígito “two” y el dígito “nine”. Siendo el desempeño del dígito “nine” el más pobre de los dos.
Por lo que se hacen dos pruebas,

- tomando todos las cadenas con un número de estados acorde a sus fonemas respectivos, excepto la cadena del dígito “nine” la cual se construye con 9 estados (N=9).
- tomando todos las cadenas con un número de estados acorde a sus fonemas respectivos, excepto la cadena del dígito “two” la cual se construye con 1 estados (N=1).

Los resultados obtenidos se resumen en la siguiente tabla:

Número de estados (N)	Precisión	Recall
Acorde al número de fonemas excepto para “nine”	0.802	0.860
Acorde al número de fonemas excepto para “two”	0.907	0.886

En la primera prueba se observa un descenso importante en la performance, en cambio sobre la segunda prueba se puede ver una mejoría en la performance global, superando incluso a la obtenida al tomar 5 estados para todas las cadenas HMM. En el primer caso la sustitución de los estados de la cadena asociada al dígito “nine” no logra la mejoría buscada para dicha cadena.
Ahora en el segundo caso si se logra el objetivo pues mientras se mantiene controlado el comportamiento del dígito “nine”, se mejora la respuesta del sistema al dígito “two”.
A partir de aquí se trabaja con la configuración de estados que globalmente produjo la mejor performance del sistema. Ya en la sección siguiente se explora como se ve afectado el sistema con el cambio en la cantidad de mixturas M.

Prueba 3

Debido a que tenemos un número finito de frames en cada instancia se podría generar algún problema si aumentamos el número de mixturas M. Como se comentó antes, en la sección anterior, para cada cadena HMM asociada a un dígito, cuando se inicializan los parámetros iniciales de la cadena en *init_mhmm.m*, el algoritmo de K-Means se encarga de agrupar los frames de todas las instancias de un mismo dígito. Separando todos estos frames en K=N*M cluster, por lo que cada

4 Las matrices de confusión se encuentran en la sección 2 del apéndice

cluster queda con un número de frames de $num\ frames_{cluster} = \frac{frames\ totales\ por\ dígito}{N \times M}$. En la práctica el número total de frames de todas las instancias de cada dígito ronda los 560 frames.

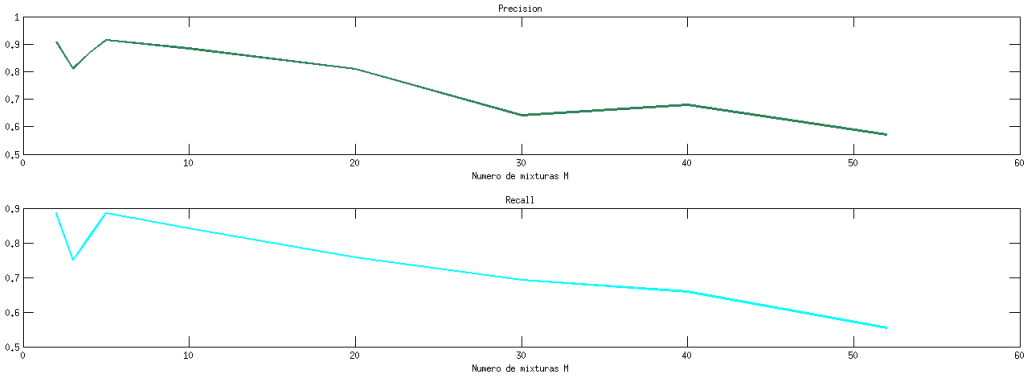
Ahora para cada gaussiana se deben estimar dos parámetros, media y varianza, por lo que es necesario que por lo menos se tenga un frame dentro de cada cluster (recordar que este frame es de dimensión igual al número de coeficientes cepstrales). Asumiendo el peor caso, trabajemos en una cadena HMM cuyo dígito asociado tiene N=6 estados, para que por lo menos exista un frame dentro de cada cluster no se debería trabajar con más de M_{max} mixturas,

$$M_{max} = \frac{frames\ totales\ por\ dígito}{N}$$

Siguiendo esta línea es claro que con un número de estados fijo (N) si aumentamos el número de mixturas gaussianas (M), disminuye el número de frames dentro de cada cluster y en consecuencia la estimación de los parámetros de cada gaussiana se vera comprometido, disminuyendo su efectividad. O sea que las gaussianas perderán la capacidad de describir apropiadamente un conjunto de frames. Produciendo en última instancia un decremento en la performance del sistema de reconocimiento de voz.

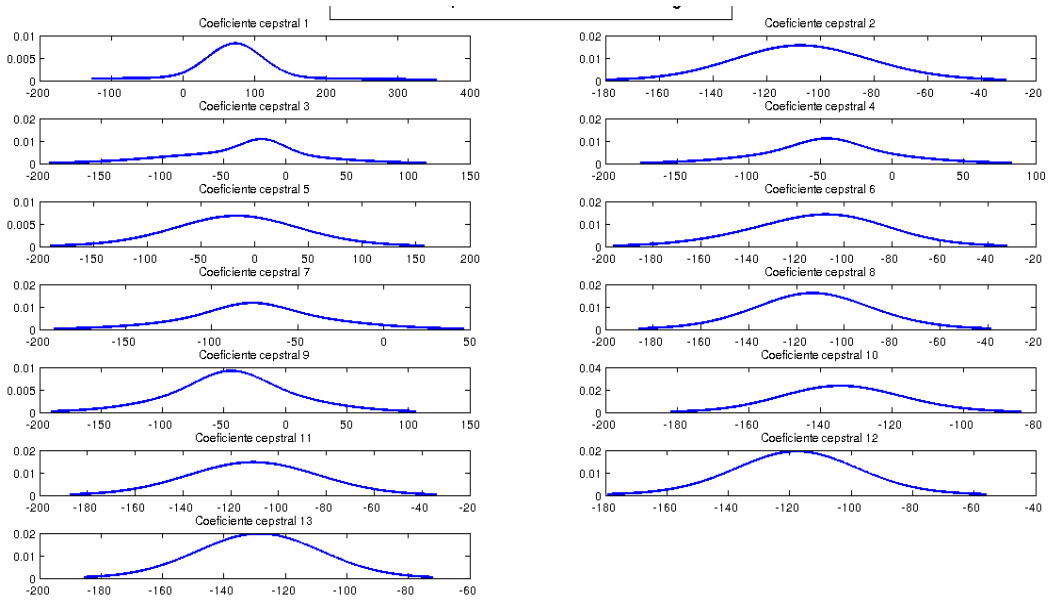
Cabe observar que el efecto de este decremento se ve reducido debido a que los códigos encargados de la etapa de inicialización y entrenamiento, no permiten que la varianza de las gaussianas este por debajo de un cierto umbral, de manera que si la varianza cae a cero las gaussianas aún sean responsables de un número razonable de puntos.

A continuación se muestra como se ven afectadas la precisión y el recall en función del número de mixturas en el modelo

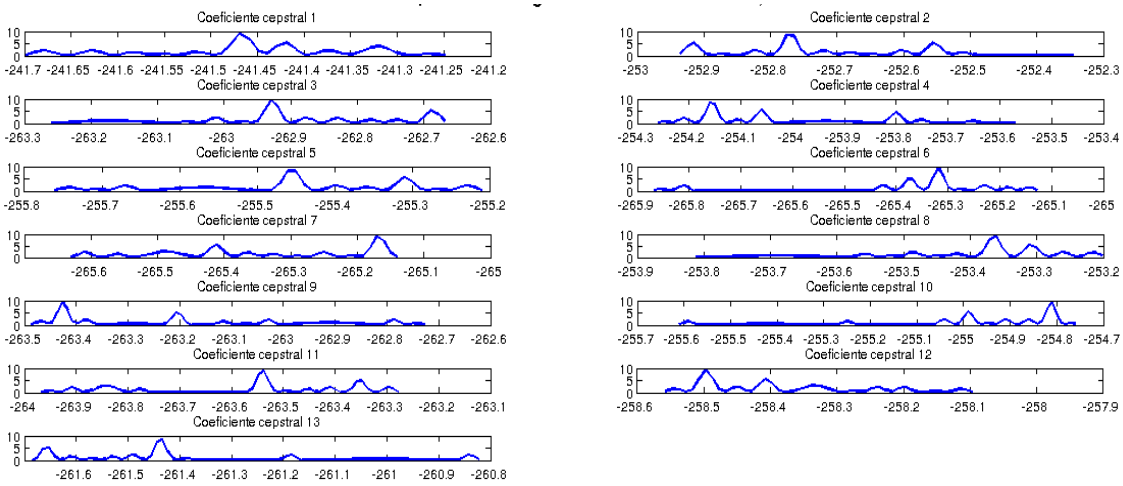


Se tiene un máximo para M=2 mixturas, produciendo una precisión de 0.907 y un recall de 0.886.

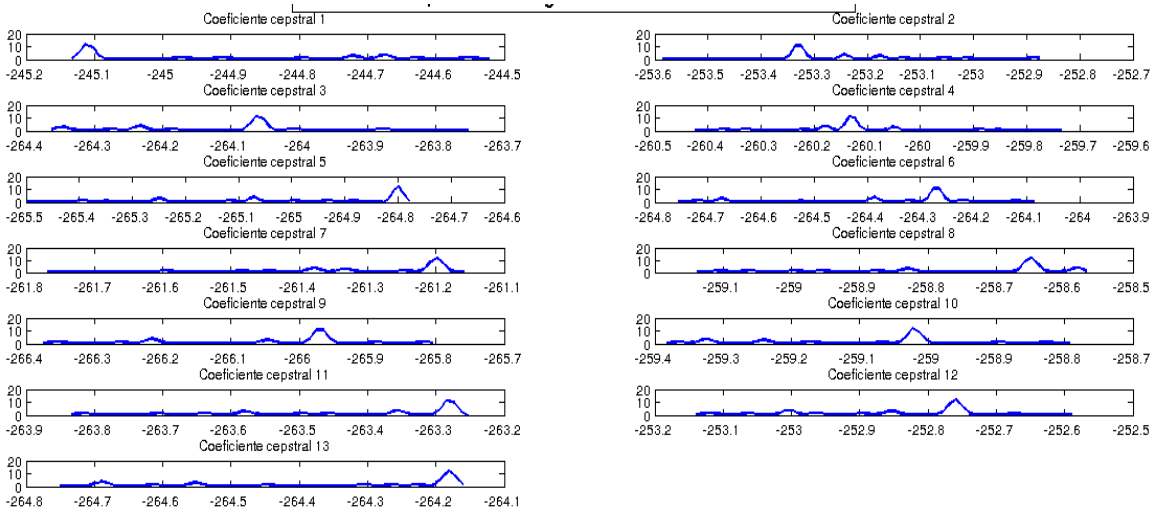
Las siguientes figuras muestran las modificaciones sobre las distribuciones asociadas a cada coeficiente cepstral al variar el número de mixturas M en la cadena HMM del dígito “zero”, solo sobre el estado 3 de esta cadena:



Distribución de probabilidad de los coeficientes cepstrales del estado 3 de la cadena asociada al dígito "zero", con M=2



Distribución de probabilidad de los coeficientes cepstrales del estado 3 de la cadena asociada al dígito "zero", con M=30



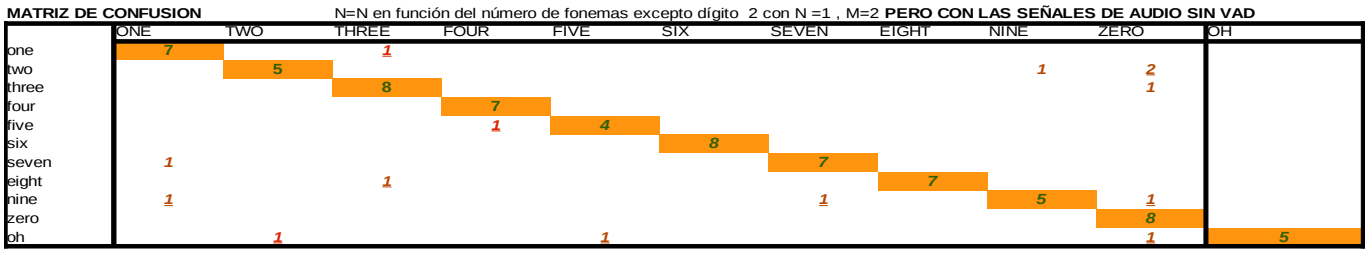
Distribución de probabilidad de los coeficientes cepstrales del estado 3 de la cadena asociada al dígito "zero", con M=52

Se puede ver el deterioro del las distribuciones asociadas a cada coeficiente cepstral, donde ya con M=30 mixturas se pierden las medias que tan buenos resultados dieron con M=2 mixturas.

Prueba 4

Esta prueba consiste en entrenar el sistema con los parámetros que han permitido el mejor desempeño, pero eliminando la etapa VAD (Voice activation detection), consistente en tomar partes útiles de los audios de cada instancia, y ver como se comporta el sistema.

A continuación se muestran las matrices de confusión en el caso con VAD, sin VAD y los respectivos valores de precisión y recall.



VAD	Precisión	Recall
Activado	0.907	0.886
Desactivado	0.855	0.835

Como es de esperar la performance de sistema baja cuando no se aplica VAD, aunque se mantiene en valores aceptables. Queda claro que si bien se contemplan los silencios iniciales y finales con estados propios en las cadenas HMM, acortar en las señales de audio, el tiempo que se está en estos estados de “silencio” antes de introducir la instancia al sistema, permite que posibles ruidos no hagan cambios de estado indeseables cuando no se tiene ninguna señal de voz, disminuyendo consecuentemente la verosimilitud del dígito a reconocer. Por lo tanto es recomendable el filtrado VAD para aumentar el desempeño del sistema.

Conclusión

Se logró integrar técnicas de extracción de características MFCC y de construcción de cadenas HMM para generar un sistema de reconocimiento de voz de palabras aisladas, capaz de reconocer dígitos en inglés con una performance global bastante satisfactoria.

Una exploración exhaustiva de las técnicas de desarrollo permitió manipular los parámetros de los toolbox utilizados y agregar algunas modificaciones con el fin de aumentar su desempeño. Tal es el caso del código VAD (Voice activation detection).

Una vez que el sistema estuvo armado se propusieron 4 pruebas para explorar cuales son los efectos de la manipulación de dos parámetros importantes, el número de estados que posee cada cadena HMM (N), y la cantidad de mixturas gaussianas (M) utilizadas para generar las distribuciones de probabilidad en cada estado.

Si bien se trabajo con una base de datos bastante reducida (solo 255 instancias en total) se obtuvieron resultados bastante aceptables.

Se pudo ver que la mayor performance global del sistema para las 88 instancias de test, se obtiene considerando un número de estados en cada cadena acorde a la cantidad de fonemas del dígito que dicha cadena intenta reconocer, exceptuando a la cadena que reconoce el dígito “two” donde se entrena la misma con solo un estado. Un número de mixturas gaussianas M= 2 para la construcción de las distribuciones de probabilidad y la utilización de VAD en las señales de entrada antes de extraer las características. La mejor precisión y recall obtenidos fue de

Número de estados (N)	Precisión	Recall
Acorde al número de fonemas excepto para “two”	0.907	0.886

APENDICE

1- Correcciones al código de Kevin Murphy para HMM

1. En el código *mhmm_em* linea 145 hay un error al llamar una función

   de

   [alpha, beta, gamma, current_loglik, xi, gamma2] = ...
   fwdback(prior, transmat, B, 'obslik2', B2, 'mixmat', mixmat);

   a

   [alpha, beta, gamma, current_loglik, xi, gamma2] = ...
   fwdback('init_state', prior, 'transmat', transmat, 'obslik', B, 'obslik2', B2, 'mixmat', mixmat);

2. en el código *fwdback* antes de la linea 51 agregar:

   compute_xi=0;
   compute_gamma2=0;

   de lo contrario no se inicializaban estas variables en algunos casos y da error.

3. Cambiar la línea 25 *de mhmm_logprob*

   de
   [alpha, beta, gamma, ll] = fwdback(prior, transmat, obslik, 'fwd_only', 1);
   a
   [alpha, beta, gamma, ll] = fwdback('init_state', prior, 'transmat', transmat,...
   'obslik', obslik, 'fwd_only', 1);

   nuevamente hay un error al llamar a la función.

2- Matrices de confusión

Notación: las mayúsculas son los las cadenas, es decir, ONE es cadena asociada a al dígito “one”

Matrices de confusión al variar el estado N de todas las cadenas, manteniendo M=2 mixturas

MATRIZ DE CONFUSION N=6 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two	1	4					3				
three			8								
four				4							
five	1			1	5						
six						8					
seven							8				
eight								8			
nine							1		5	2	
zero										8	
oh	2										6

MATRIZ DE CONFUSION N=5 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		3								4	
three			8								
four				8							
five				1	7						
six						8					
seven							8				
eight								8			
nine	1						2		2	3	
zero										8	
oh				2							6

MATRIZ DE CONFUSION N=4 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		5					3				
three			5			2		1			
four				6						1	1
five					7						1
six						8					
seven							8				
eight								8			
nine	3						4		0	1	
zero							1			7	
oh					1						7

MATRIZ DE CONFUSION N=3 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		7								1	
three			8								
four				8							
five				1	7						
six						8					
seven							7			1	
eight			1					6		1	
nine	3						2		1	2	
zero					5					7	
oh				1							7

MATRIZ DE CONFUSION N=2 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	2								4		2
two		7					1				
three			7					1			
four				8							
five				1	7						
six						8					
seven							8				
eight			1					7			
nine	1						2		4	1	
zero									1	9	
oh											8

MATRIZ DE CONFUSION N=1 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		8									
three			7					1			
four				7							1
five				1	7						
six						8					
seven	1						7				
eight			1					7			
nine	2						1		2	3	
zero										8	
oh											8

MATRIZ DE CONFUSION N=7 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		2					4			2	
three			5				1	2			
four				8							
five				1	6					1	
six						8					
seven							8				
eight						2		6			
nine							1		4	3	
zero										8	
oh				3							5

MATRIZ DE CONFUSION N=8 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two	1	7									
three			7							1	
four				8							
five				1	7						
six					1	7					
seven	2				4	2	0				
eight		1						4		3	
nine	5				1				1		
zero										8	
oh				2							6

MATRIZ DE CONFUSION N=9 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	4		2	2							
two		3	1			1			1	2	
three			6					2			
four				8							
five				1	8						
six						8					
seven			1				7				
eight								8			
nine				1			1		5	1	
zero										8	
oh				2							6

MATRIZ DE CONFUSION N=10 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	2		2				1		3		
two		6					2				
three			7			1					
four				8							
five				1	7						
six						8					
seven						1	6		1		
eight			1			2		5			
nine	1						2		4	1	
zero										8	
oh									1		7

Matrices de confusión para el sistema de reconocimiento de voz utilizando para cada cadena un número de estados acorde con la cantidad de fonemas del dígito asociado.

MATRIZ DE CONFUSION N en función del número de fonemas, M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		3								5	
three			7					1			
four				8							
five				1	7						
six						8					
seven							8				
eight								8			
nine	3						2		1	2	
zero										8	
oh											8

Utilizando algunas cadenas con un número arbitrario de estados y el resto con un número de estados acorde con la cantidad de fonemas del dígito asociado.

MATRIZ DE CONFUSION N en función del número de fonemas excepto dígito 9 con N =9 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		8									
three			8								
four				7							1
five				1	5						
six						8					
seven	1						7				
eight			1					7			
nine	2						2		0	4	
zero										8	
oh											8

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=2

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		7					1				
three			7					1			
four				8							
five				1	7						
six						8					
seven							8				
eight								8			
nine	2						2		3	1	
zero										8	
oh		2									6

Matrices de confusión al variar el número de mixturas gaussianas (M)

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=3

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	7									1	
two	1	6								1	
three			6					2			
four				6						1	1
five				1	7						
six						8					
seven							7			1	
eight						3		3		2	
nine	1						2		2	3	
zero										8	
oh	2										6

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=4

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two	1	7									
three			6					2			
four				8							
five				1	7						
six						8					
seven							7				
eight						2		6			
nine	4						1		1	2	
zero										8	
oh										3	5

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 ,, M=5

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		7								1	
three		1	6					1			
four				8							
five				1	7						
six						8					
seven							7				
eight								8			
nine	1								3	4	
zero										8	
oh				1							7

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=10

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		5				1				2	
three			8								
four	1			6						1	
five				1	7						
six						8					
seven							7			1	
eight								8			
nine	4				1		1		2		
zero										8	
oh										1	7

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=20

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	7		1								
two		4	2							2	
three			8								
four				5	1					1	
five				1	6					1	
six						8					
seven							7		1		
eight								8			
nine	3		1				2		2		
zero										8	
oh					1					4	3

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=40

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	8										
two		0	7				1				
three			7							1	
four				8							
five				2	6						
six						7	1				
seven			1				7				
eight			2			2		2	2		
nine	4						1		2	1	
zero										8	
oh	1	2		2							3

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=52

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	0		1						5	2	
two		0				1			1	6	
three			3		1			2		2	
four				2	1	1			1	3	
five					7					1	
six						8					
seven						2	5				
eight						1		3		4	
nine					1				6	1	
zero					1					7	
oh					1						7

MATRIZ DE CONFUSION N=N en función del número de fonemas excepto dígito 2 con N =1 , M=30

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO	OH
one	7									1	
two		0					1		3	4	
three			8							4	
four				4							
five				1	7						
six						6	2				
seven							8				
eight			1					6		1	
nine	22			1			2		2	1	
zero										8	
oh				2						1	5

FUENTES

- Rabiner, L. & Juang, B-H.. Fundamentals of Speech Recognition, Prentice Hall, N.J., 1993.
- Rabiner, L.. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings of the IEEE, Vol. 77, No. 2, pp: 257-286, 1989.
- Código de Dan Ellis “PLP and RASTA(and MFCC, and inversion) in Matlab” http://labrosa.ee.columbia.edu/matlab/rastamat/
- Diapositivas de Dan Ellis EEE6820: Speech & Audio Processing & Recognition-Lecture 11
- K.R. Aida-Zade, C. Ardil and S.S. Rustamov - Investigation of Combined use of MFCC and LPC Features in Speech Recognition Systems
- Toolbox que permite la implementación en Matlab de los HMM para reconocimiento de voz es el provisto por Kevin Murphy[5] , “Hidden Markov Model (HMM) Toolbox for Matlab”. http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
- Materiales del curso ProDiVoz de la *Facultad de Ciencias Exactas, Ingeniería y Agrimensura,* Argentina http://www.fceia.unr.edu.ar/prodivoz/programa_index.html

5 http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html . El código presenta algunos errores, ver el apéndice para la corrección de los mismos.