Procesamiento Digital de Señales de Voz

Guía para la implementación de HMM basado en Matlab

Octubre 2007

Los Modelos Ocultos de Markov (*Hidden Harkov 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 o_t en el instante *t*. Puede pensarse a un modelo HMM como una máquina de estados finitos en la cual cada estado tiene asociada una Función de Densidad de Probabilidad (FDP) para cada vector característico.

Los parámetros que definen un HMM son:

1. *El número de estados N del modelo*. Generalmente los estados se interconectan de manera que cualquier estado puede alcanzarse desde otro estado (modelos ***ergódicos***) . El conjunto de todos los estados posibles se denota:

$$S = \{S_1, S_2, \ldots, S_N\}$$

2. *El número de símbolos de observación posibles M para cada estado*. Estos símbolos se corresponden con las salidas físicas del sistema a modelar y se los denota:

$$V = \{V_1, V_2, \ldots, V_M\}$$

3. *La distribución de probabilidades de transición de un estado a otro estado.* Indica la probabilidad de que estando el sistema en un estado, en un instante determinado, el mismo evolucione a otro estado o permanezca en el mismo estado, en el instante subsiguiente. Lo detonamos con:

$$A = \{a_{ij}\}$$

$$a_{ij} = P\left(q_{t+1} = S_j / q_t = S_i\right) \qquad 1 \le i, j \le N$$

4. *La distribución de probabilidades de observación de un símbolo en el estado j*

$$B = \{b_j(k)\} = \left\{P\left(V_k \text{ en } t / q_t = S_j\right)\right\} \qquad 1 \le j \le N$$

5. *La distribución inicial de probabilidades de cada estado.*

$$\pi = \{\pi_i\}$$

$$\pi_i = P\left(q_1 = S_i\right) \qquad 1 \le i \le N$$

Así, una manera compacta de representar un modelo HMM es la siguiente

$$\lambda = (A, B, \pi) \tag{1}$$

En las aplicaciones de HMM en reconocimiento de voz es necesario calcular la probabilidad de que una secuencia de observaciones ***O*** haya sido generada por el modelo λ , i.e. $P[\boldsymbol{O} \mid \lambda]$.

En particular, en el modelado de clasificadores utilizados para reconocimiento de voz, es conveniente que las observaciones en cada estado (vectores característicos) presenten un comportamiento continuo. En este caso, la distribución de probabilidad *B* se convierte en una función de densidad de probabilidad, que puede representarse por una combinación lineal de distribuciones Gaussianas (mezclas Gaussianas), es decir

$$b_j(O) = \sum_{k=1}^{M} c_{jk}\, N\left(O, \boldsymbol{\mu}_{jk}, \boldsymbol{U}_{jk}\right) \qquad 1 \le j \le N \tag{2}$$

donde c_{jk} es el coeficiente para la *k*-ésima mezcla en el estado *j* y *O* es el vector de observaciones a ser modelado. *N* es una distribución Gaussiana con media $\boldsymbol{\mu}_{jk}$ y $\boldsymbol{U}_{jk}$ es la matriz de covarianza para *k*-ésima muestra en el estado *j*. Cabe destacar que la FDP de la ecuación (2) se puede utilizar para aproximar cualquier función de densidad continua y finita.

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 está disponible en el sitio web de la asignatura[1]. El mismo surge como una modificación del ***Hidden Markov Model (HMM) Toolbox for Matlab*** desarrollado por *Kevin Murph*[2].

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

1. Inicialización: `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, ver Figura 1. 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.

[1] http://www.fceia.unr.edu.ar/prodivoz/trabajos_practicos_index.htm
[2] El Toolbox está disponible en: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

`B:` Matriz de distribución de probabilidades de observación.

`Mu:` Matriz con los valores medios asociados a las distribuciones Gaussianas.

`Sigma:` Matriz con los desvíos estándar asociados a las distribuciones Gaussianas.

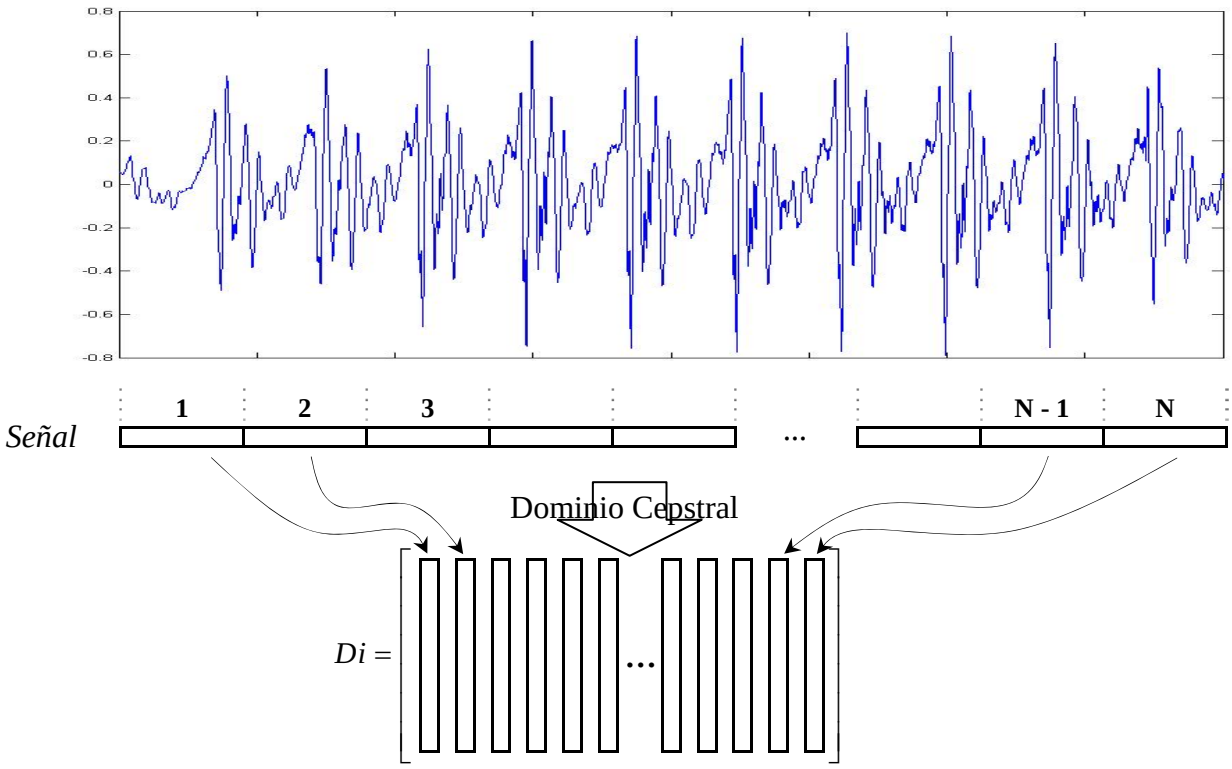

Figura 1: Estructura de la matriz `Di`. La i-ésima columna de la matriz `Di` contiene los coeficientes cepstral del i-ésimo frame

Cuando se utilizan los modelos HMM para el reconocimiento de señales de voz no todas las transiciones de estado son posibles, resultando en un modelo denominado de izquierda-a-derecha. Por tal motivo, es necesario restringir las estructuras de la matriz de transición y de la matriz de probabilidad inicial. Así, se incluyeron las siguientes líneas en el código de la función `init_mhmm.m`.

```
transmat = mk_leftright_transmat(Q,0.5);
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 y la segunda línea impone que la secuencia de estados comience en el estado 1.

Los valores recomendados de los parámetros de entrada para el reconocimiento de dígitos son:

```
2 < N < 6
3 < M < 12
cov_type = 'full' o 'diag'
```

2. 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)
   ```

   Nota: Si bien estos parámetros son opcionales, la función mostrará un mensaje de error si no es llamada con al menos uno de ellos.

   Salidas:

   `pi, A, mu, sigma, B:` Matrices con los valores resultantes del entrenamiento del modelo.

   `ll_trace:` Vector que indica, en cada iteración del entrenamiento, cual es la probabilidad de que los datos de entrenamiento hayan sido generados por el modelo. Este parámetro guarda información asociada al entrenamiento y no es necesario para la etapa de reconocimiento.

3. Reconocimiento: `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.