Trabajo Práctico
Modelos Ocultos De Markov

Anastópulos Matías - 95120

2 de junio de 2018

Índice

1. Introducción **1**

2. Estimación De Parámetros Del Modelo **2**
2.1. Recursión Foward Backward 2
2.1.1. Código 2
2.2. Expectación Maximización (EM) 4
2.2.1. Resultados 4
2.2.2. Código 8

3. Detección De Secuencias De Palabras **11**
3.1. Matrices Concatenadas 12
3.2. Resultados 12
3.3. Código 13

4. Conclusiones **15**

Resumen

El objetivo de este trabajo es la implementación de algunos algoritmos relacionados con modelos ocultos de Markov con el fin de lograr un mayor entendimiento de los mismos. Realizaremos una estimación de parámetros de los mismos mediante EM. Luego, dados dos modelos ocultos de Marcov, realizaremos emisiones aleatorias de los mismos en forma concatenada. El objetivo será reconocer mediante el algoritmo de Viterbi, que secuencia de modelos fue la que generó dicha emisión. Esto esta relacionado con la detección de secuencias de palabras.

1. Introducción

Un modelo oculto de Markov es una maquina de estados que recorre una secuencia de estados $\{q_1, q_2, ..., q_T\}$ de manera aleatoria. En cada estado se realiza una emisión de un valor, de forma que obtenemos una secuencia $\{x_1, x_2, ..., x_T\}$. En el modelo tomamos dos hipótesis:

$$P(q_t|q_{t-1}, ..., q_1) = P(q_t|q_{t-1}) \tag{1}$$

O sea la transición al estado siguiente depende únicamente del estado actual.

$$P(x_t|q_t, ..., q_1, x_{t-1}, ..., x_1) = P(x_t|q_t) \tag{2}$$

O sea la emisión depende únicamente del estado actual. De esta forma el modelo queda determinado por los siguientes parámetros:

$$a_{i,j} = P(q_t|q_{t-1}) \tag{3}$$

Que es la matriz de transición.

$$b_j(x_t) = P(x_t|q_t) \tag{4}$$

Que es la distribución de la emisión en un determinado estado. En general se toman como normales o mezclas de normales. De esta forma tendrán matrices de covarianza y medias que deberán ser estimadas. En general en procesamiento del habla tendremos un estado inicial y uno final que no realizan emisiones.

2. Estimación De Parámetros Del Modelo

El objetivo de esta sección es, dada una realización de un modelo oculto de Markov, poder estimar los parámetros del mismo. Necesitamos estimar la matriz de transición, las medias y varianzas de las normales de cada estado. Esto lo realizaremos mediante el algoritmo EM.

2.1. Recursión Foward Backward

Para la implementación del algoritmo EM necesitamos disponer de un algoritmo que haga la recursión foward-backward, obteniendo las matrices alfa, beta, gamma y xi.

2.1.1. Código

A continuación incluimos el código que realiza el calculo de las matrices mediante la recursión foward-backward.

```
function [alphas, betas, gammas, xis, logProbAlpha, logProbBeta] = logalphabeta(x,means,
    vars,transitions)

if nargin == 2,
  model = means;
  means = model.means;
  vars = model.vars;
  model.trans(model.trans<1e-100) = 1e-100;
  logTrans = log(model.trans);
end;

numStates = length(means);
nMinOne = numStates - 1;
[numPts,dim] = size(x);

log2pi = log(2*pi);
for i=2:nMinOne,
  invSig{i} = inv(vars{i});
  logDetVars2(i) = -0.5 * log(det(vars{i})) - log2pi;
end;

%Initialize the alpha vector for the emitting states
for i=2:nMinOne,
  X = x(1,:)-means{i}';
  alpha(i) = logTrans(1,i) ...
      - 0.5 * (X * invSig{i}) * X' + logDetVars2(i);
end;
alpha = alpha(:);

alphas = zeros(length(alpha),numPts);
alphas(:, 1) = alpha(:);

%Do the forward recursion
for t = 2:numPts
  alphaBefore = alpha;
  for i = 2:nMinOne
    X = x(t,:)-means{i}';
    alpha(i) = logsum( alphaBefore(2:nMinOne) + logTrans(2:nMinOne,i) ) ...
        - 0.5 * (X * invSig{i}) * X' + logDetVars2(i);
  end;
  alphas(:, t) = alpha(:);
end;

%Terminate the recursion with the final state
logProb =  logsum( alpha(2:nMinOne) + logTrans(2:nMinOne,numStates) );

logProbAlpha = logProb;

%Inicializo vector de betas.
```

```
beta = logTrans(1:nMinOne, end);

betas = zeros(length(beta),numPts);
betas(:, numPts) = beta(:);
%betas
% Recursion Backward
t = numPts - 1;
while(t >= 1)
    betaBefore = beta;
    for i = 2:nMinOne
        b = zeros(1, nMinOne);
        for j = 2:nMinOne
            X = x((t + 1),:) - means{j}';
            b(j) = - 0.5 * (X * invSig{j}) * X' + logDetVars2(j);
        end
        beta(i) = logsum(betaBefore(2:nMinOne)' + logTrans(i, 2:nMinOne) ...
            + b(2:nMinOne));
    end
    betas(:, t) =  beta(:);
    t = t - 1;
end

b = zeros(1, nMinOne);
for j = 2:nMinOne
  X = x(1,:) - means{j}';
  b(j) = - 0.5 * (X * invSig{j}) * X' + logDetVars2(j);
end

logProbBeta = logsum(b(2:nMinOne) + betas(2:nMinOne, 1)' + logTrans(1, 2:nMinOne));

[N, T] = size(alphas);
gammas = zeros(N, T);
for t = 1:T
  for k = 2:N
        gammas(k, t) = alphas(k, t) + betas(k, t) - ...
            logsum(alphas(2:N, t) + betas(2:N, t));
  end
end

xis = zeros(N, N, T);
for t = 2:T
    for j = 2:N
        for k = 2:N
            X = x(t, :) - means{k}';
            b = - 0.5 * (X * invSig{k}) * X' + logDetVars2(k);
            %xis(j, k, t) = alphas(j, t - 1) + b + logTrans(j, k) + ...
            %    betas(k, t) - logsum(gammas(2:N, t));
            xis(j, k, t) = alphas(j, t - 1) + b + logTrans(j, k) + ...
                betas(k, t) - logsum(alphas(2:N, t) + betas(2:N, t));

        end
    end
end

%=====================================
function result = logsum(logv)

len = length(logv);
if (len<2);
  error('Subroutine logsum cannot sum less than 2 terms.');
end;

% First two terms
if (logv(2)<logv(1)),
  result = logv(1) + log( 1 + exp( logv(2)-logv(1) ) );
else,
  result = logv(2) + log( 1 + exp( logv(1)-logv(2) ) );
end;

% Remaining terms
for (i=3:len),
  term = logv(i);
  if (result<term),
    result = term   + log( 1 + exp( result-term ) );
```

```
123   else,
        result = result + log( 1 + exp( term-result ) );
125   end;
end;
```

codigo/calcular_matrices.m

2.2. Expectación Maximización (EM)

El algoritmo EM es iterativo, el objetivo será, dada una realización y una serie de parámetros iniciales, obtener las matrices alfa, beta, gamma y xi. Luego, con las mismas, volver a estimar los parámetros. Una vez estimados se podrá volver a calcular las matrices. Se deberá repetir el ciclo hasta que el *likelihood* de la realización ya no varíe demasiado.

2.2.1. Resultados

En las siguientes figuras podemos ver como la implementación del algoritmo evoluciona en cada iteración. A medida que transcurre el ciclo, las medias y varianzas se acomodan a los datos de la realización.

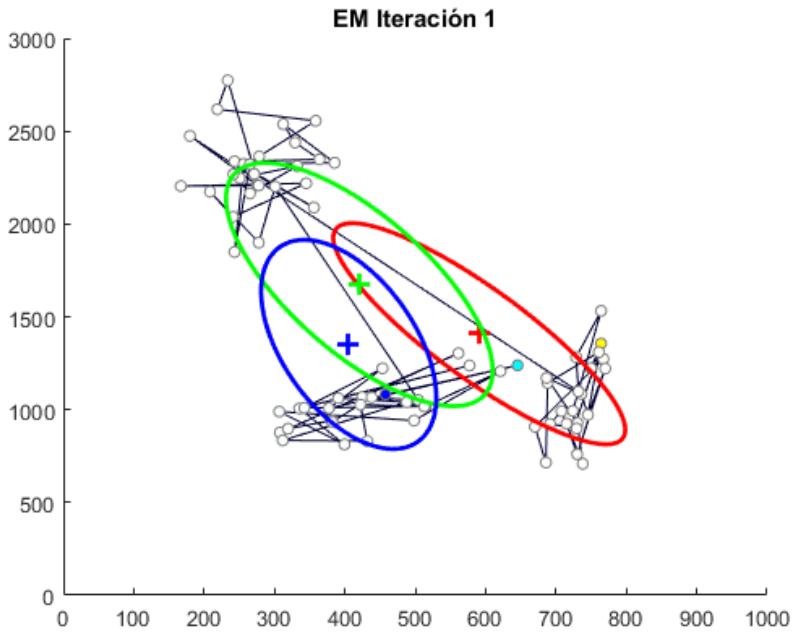

Figura 1: Iteración 1

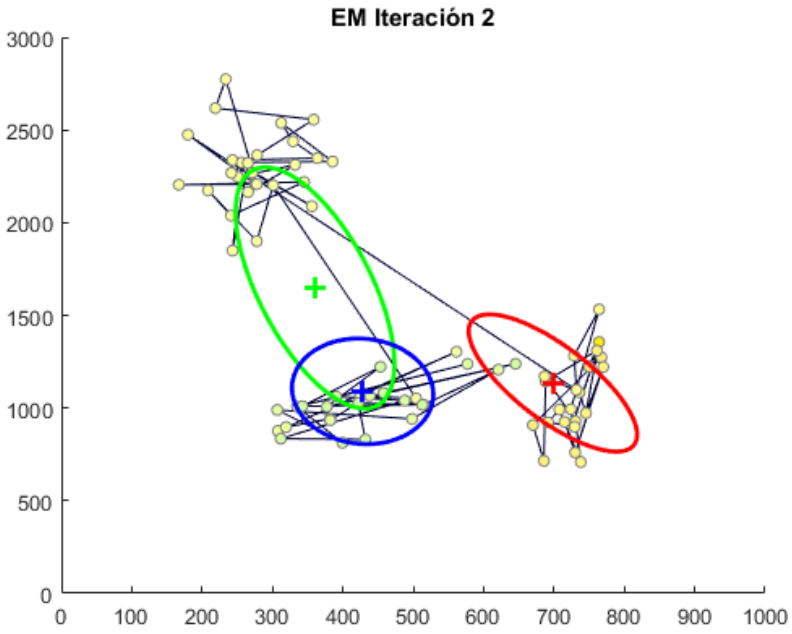

Figura 2: Iteración 2

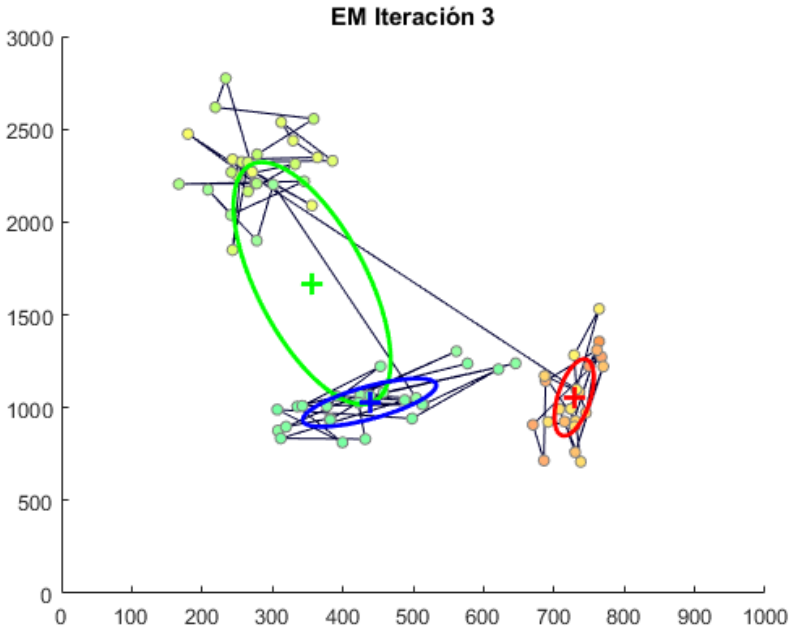

Figura 3: Iteración 3

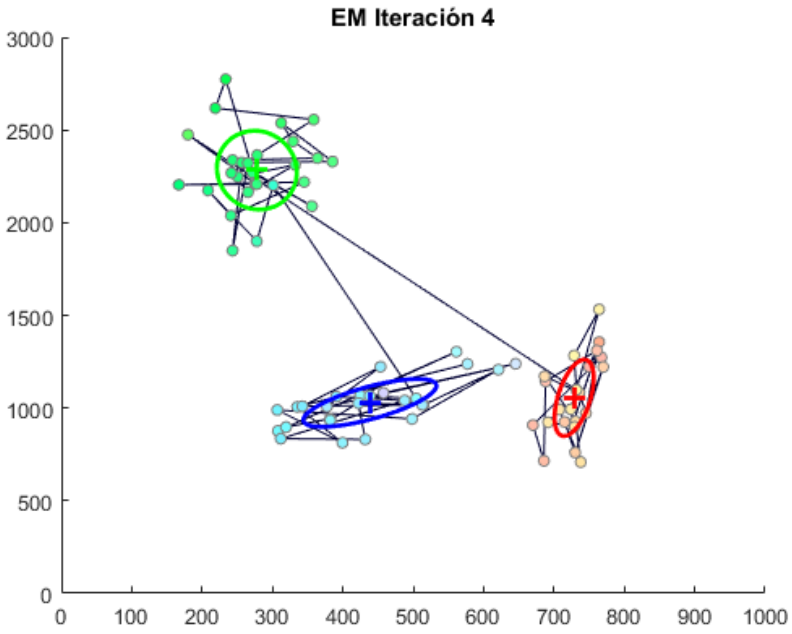

Figura 4: Iteración 4

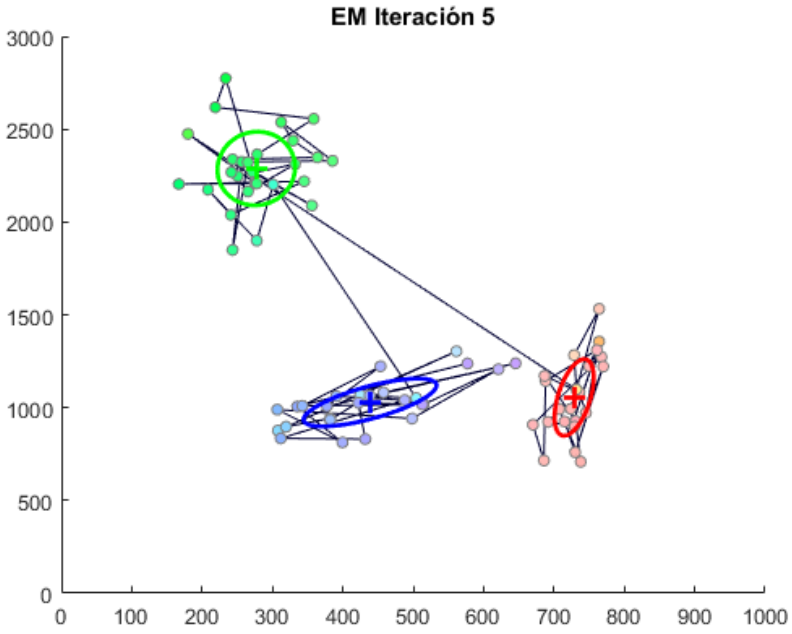

Figura 5: Iteración 5

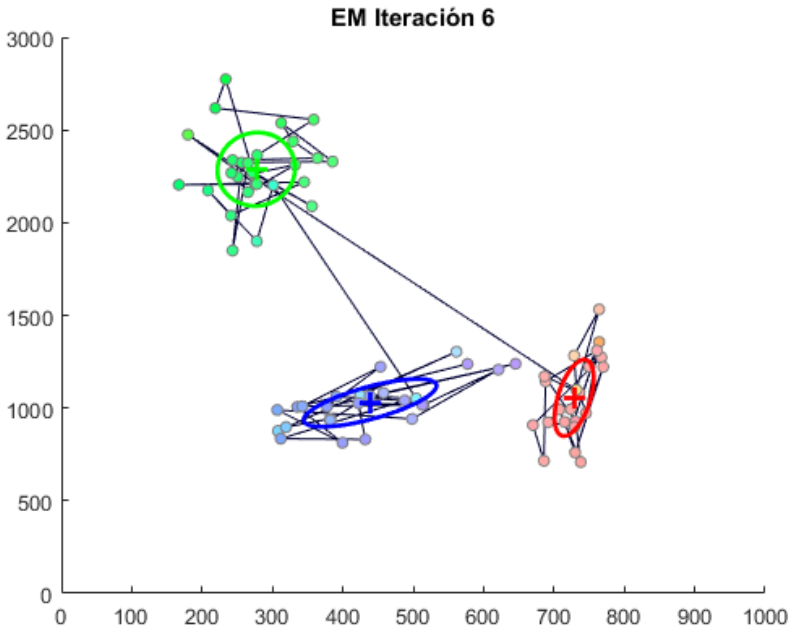

Figura 6: Iteración 6

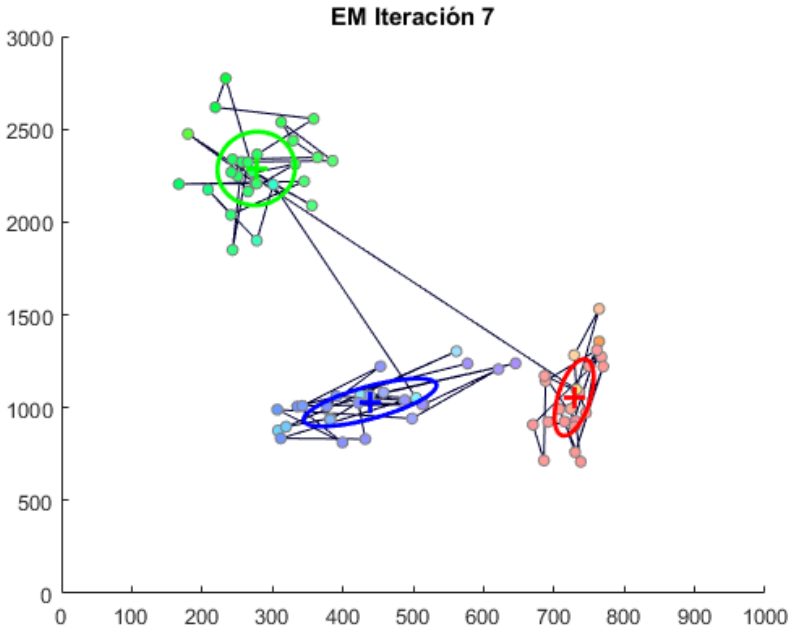

Figura 7: Iteración 7

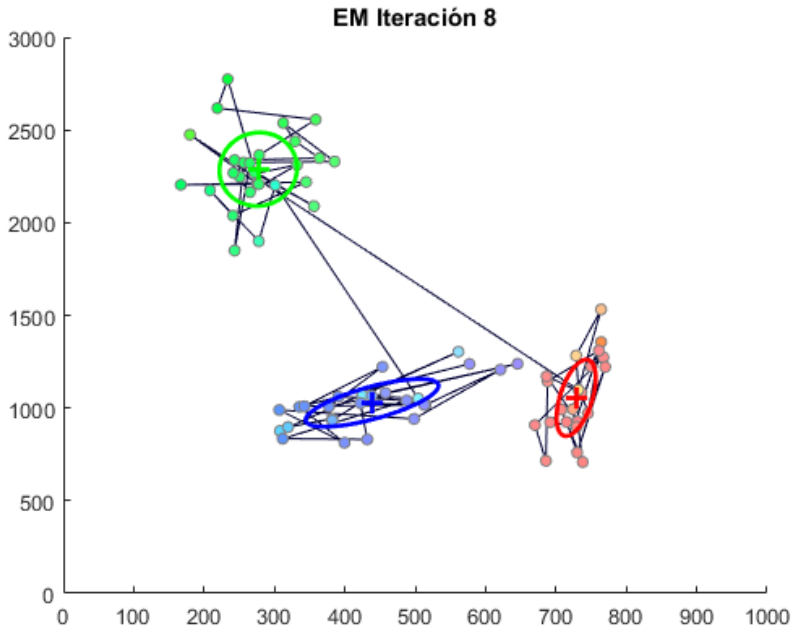

Figura 8: Iteración 8

En la figura 9 podemos ver la evolución del *likelihood* en cada iteración.

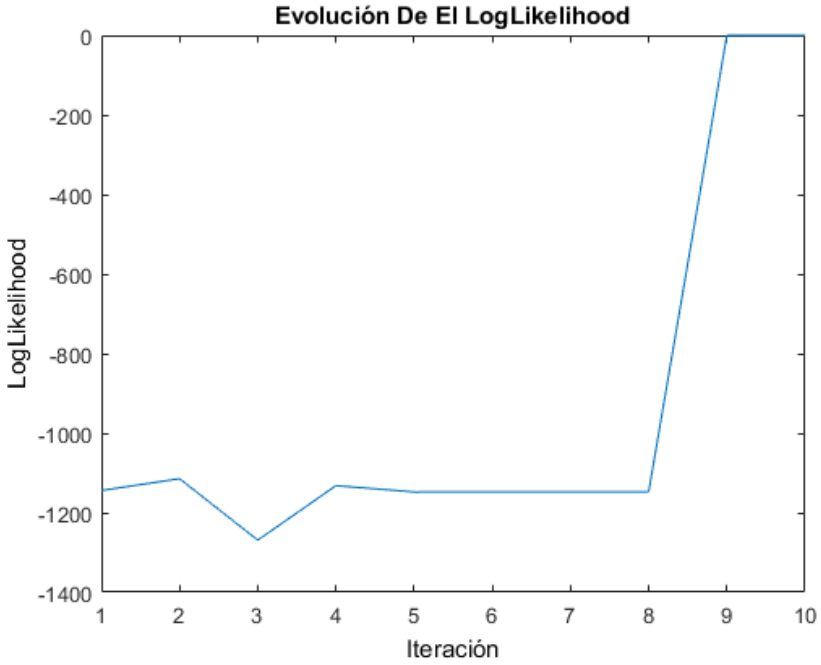

Figura 9: Evolución Del Likelihood

2.2.2. Código

A continuación incluimos el código que realiza la estimación mediante EM.

```
% Inicializacion.
close all
clear
clc
colordef white

```

```
% Cargo Datos
load data;

% Parametros.
MINIMO_REALIZACIONES_DE_ESTADO = 15;
ITERACIONES_MAXIMAS = 10;
MAXIMO_NUMERO_DE_CONDICION = 5;

% Genero la realizacion del HMM, itero hasta obtener una con suficientes
% puntos en cada estado.
data_ok = 0;
while(data_ok == 0)
    [X, Q] = genhmm(hmm4);
    states_1 = sum(Q(:) == 2);
    states_2 = sum(Q(:) == 3);
    states_3 = sum(Q(:) == 4);
    if((states_1 > MINIMO_REALIZACIONES_DE_ESTADO) ...
            && (states_2 > MINIMO_REALIZACIONES_DE_ESTADO) ...
            && (states_3 > MINIMO_REALIZACIONES_DE_ESTADO))
        data_ok = 1;
    end
end

% Calculo las medias, varianzas y matriz de transición iniciales.
mu_inicial = media(X);
sigma_inicial = sigma(X, mu_inicial);
mu_1 = mu_inicial;
mu_2 = mu_inicial;
mu_3 = mu_inicial;
sigma_1 = sigma_inicial;
sigma_2 = sigma_inicial;
sigma_3 = sigma_inicial;
A = [0,   1,   0,   0,   0; ...
     0, 0.5, 0.5,   0,   0; ...
     0,   0, 0.5, 0.5,   0; ...
     0,   0,   0, 0.5, 0.5; ...
     0,   0,   0,   0,   1];

% Iteracion EM.

likelihoods = zeros(1, ITERACIONES_MAXIMAS);
delta_likelihood = Inf;
iter = 1;

while((iter <= ITERACIONES_MAXIMAS) && (delta_likelihood > 0.01))
    % Calculo las matrices con las probabilidades alfa, beta, gamma y xi.
    mi_hmm.means = {[], mu_1', mu_2', mu_3', []};
    mi_hmm.vars  = {[], sigma_1, sigma_2, sigma_3, []};
    mi_hmm.trans = A;

    [alphas, betas, gammas, xis, logpa, logpb] = calcular_matrices(X, mi_hmm);

    % Recalculo parametros.
    T = length(X);
    mu_1 = estimador_mu(X, 2, exp(gammas));
    mu_2 = estimador_mu(X, 3, exp(gammas));
    mu_3 = estimador_mu(X, 4, exp(gammas));

    sigma_1_bis = estimador_sigma(X, mu_1, 2, exp(gammas));
    sigma_2_bis = estimador_sigma(X, mu_2, 3, exp(gammas));
    sigma_3_bis = estimador_sigma(X, mu_3, 4, exp(gammas));

    if(rcond(sigma_1_bis) < MAXIMO_NUMERO_DE_CONDICION)
        sigma_1 = sigma_1_bis;
    end
    if(rcond(sigma_2_bis) < MAXIMO_NUMERO_DE_CONDICION)
        sigma_2 = sigma_2_bis;
    end
    if(rcond(sigma_3_bis) < MAXIMO_NUMERO_DE_CONDICION)
        sigma_3 = sigma_3_bis;
    end

    A = estimador_transicion(exp(xis), T);

```

```
    %Figuras
82  figure(iter)
    plotseq3(X, gammas, iter);
84  plotgaus(mu_1, sigma_1, [1, 0, 0]);
    plotgaus(mu_2, sigma_2, [0, 1, 0]);
86  plotgaus(mu_3, sigma_3, [0, 0, 1]);
    axis([0, 1000, 0, 3000]);
88
    %Calculo cuanto varió el likelihood, para saber si cortar.
90  likelihoods(iter) = logpa;
    if(iter == 1)
92      delta_likelihood = Inf;
    elseif(likelihoods(iter) < likelihoods(iter - 1))
94      delta_likelihood = Inf;
    else
96      delta_likelihood = likelihoods(iter) - likelihoods(iter - 1);
    end
98
    iter = iter + 1;
100 end

102 %Figura de evolución del likelihood.
    figure(iter);
104 plot(likelihoods);
    xlabel('Iteración')
106 ylabel('LogLikelihood')
    title('Evolución De El LogLikelihood');
```

codigo/ejercicio_1.m

Al inicializar los parámetros se utilizó la media y varianzas globales. El siguiente código es el que se utilizó para esto:

```
1  function resultado_media = media(x_n)
       [N, d] = size(x_n);
3      suma = zeros(1, d);
       for i = 1:N
5          suma = suma + x_n(i,:);
       end
7      resultado_media = suma / N;
   end
```

codigo/media.m

```
   function resultado_sigma = sigma(x_n, mu)
2      [N, d] = size(x_n);
       suma = zeros(d, d);
4      for i = 1:N
           suma = suma + (x_n(i,:) - mu)' * (x_n(i,:) - mu);
6      end
       resultado_sigma = suma / N;
8  end
```

codigo/sigma.m

Para la estimación de las medias, varianzas y matriz de transición en cada iteración se utilizaron los siguientes archivos:

```
   function mu = estimador_mu(x, clase, gammas)
2      T = length(x);
       mu = zeros(1,2);
4      for t = 1:T
           mu = mu + x(t, :) * gammas(clase, t);
6      end
       mu = mu / sum(gammas(clase,1:T));
8  end
```

codigo/estimador_mu.m

```
function sigma = estimador_sigma(x, mu, clase, gammas)
    T = length(x);
    sigma = zeros(2, 2);
    for t = 1:T
        sigma = sigma + (x(t, :) - mu)' * (x(t, :) - mu) * gammas(clase, t);
    end
    sigma = sigma / sum(gammas(clase,1:T));
end
```

codigo/estimador_sigma.m

```
function A = estimador_transicion(xis, T)
    A = zeros(5, 5);
    for j = 2:4
        for k = 2:4
            A(j,k) = 0;
            for t = 2:T
                A(j,k) = A(j,k) + xis(j, k, t);
            end
            denominador = 0;
            for t = 2:T
                for k2 = 2:4
                    denominador = denominador + xis(j, k2, t);
                end
            end
            A(j, k) = A(j, k) / denominador;
        end
    end
    A(2,5) = 1 - A(2, 4) - A(2, 3) - A(2, 2);
    A(3,5) = 1 - A(3, 4) - A(3, 3) - A(3, 2);
    A(4,5) = 1 - A(4, 4) - A(4, 3) - A(4, 2);
    A(1,:) = [0, 1, 0, 0, 0];
    A(5,:) = [0, 0, 0, 0, 1];
end
```

codigo/estimador_transicion.m

Finalmente para realizar los gráficos se utilizó la siguiente función:

```
function [hy,h] = plotseq3(x, gammas, iter)

T = length(x);

hold on;

set(gca, 'color', 'w');

plot(x(:,1), x(:,2), 'color', [0, 0, 0.2]);

for t = 1:T
    gammas = gammas + abs(min(min(gammas)));
    gammas = gammas / max(max(gammas));
    color = [gammas(2, t), gammas(3, t), gammas(4, t)];
    plot(x(t, 1), x(t, 2), 'color', [0.5, 0.5, 0.5], 'marker','o','markerface', color, '
    markerSize', 5, ...
      'linestyle','none');
end

title(sprintf('EM Iteración %d', iter));
```

codigo/plotseq3.m

3. Detección De Secuencias De Palabras

En esta sección lo primero que realizaremos será la unión de dos modelos ocultos de Markov, de forma que una realización de la mezcla inicie por cualquiera de los dos. Una vez finalizada la secuencia de uno en particular, la realización podrá reiniciarse en cualquiera de los dos modelos o pasar al estado final, terminando la realización. Luego generaremos realizaciones de la unión de ambos modelos. Mediante

el algoritmo de Viterbi, obtendremos la secuencia de estados optima para esa realización y finalmente mediante una maquina de estados obtendremos la secuencia de sub-modelos que recorrió la realización.

3.1. Matrices Concatenadas

En la figure 10 podemos ver ambas matrices de ambos modelos, en rojo tenemos las probabilidades de terminar de cada modelo.

0	1	0	0	0
0	A(2,2)	A(2,3)	0	0
0	0	A(3,3)	A(3,4)	0
0	0	0	A(4,4)	A(4,5)
0	0	0	0	1

0	1	0	0	0
0	B(2,2)	B(2,3)	0	0
0	0	B(3,3)	B(3,4)	0
0	0	0	B(4,4)	B(4,5)
0	0	0	0	1

Figura 10: Matrices De Ambos Modelos

En la figura 11 se puede ver la concatenación de ambas matrices. En rojo vemos las probabilidades de reiniciar en un modelo o en otro, o directamente terminar.

0	0.5	0	0	0.5	0	0	0
0	A(2,2)	A(2,3)	0	0	0	0	0
0	0	A(3,3)	A(3,4)	0	0	0	0
0	A(4,5)/3	0	A(4,4)	A(4,5)/3	0	0	A(4,5)/3
0	0	0	0	B(2,2)	B(2,3)	0	0
0	0	0	0	0	B(3,3)	B(3,4)	0
0	B(4,5)/3	0	0	B(4,5)/3	0	B(4,4)	B(4,5)/3
0	0	0	0	0	0	0	1

Figura 11: Matrices Concatenadas

3.2. Resultados

A continuación presentamos el resultado del algoritmo implementado. Puede verse que la probabilidad de la secuencia generada es casi igual a la probabilidad de la secuencia de estados optima. Finalmente nos dice la secuencia de sub-modelos que recorrió y la secuencia de estados real de esa realización.

```
Probabilidad (Log): -1897.255634
Probabilidad Secuencia Optima (Log): -1897.264618
La secuencia fue:
secuencia_hmm =

     4     4     6

>> q

q =

  Columns 1 through 20

    1     2     2     2     2     2     2     2     2     2     2     2     2     2     2
    2     3     3     3     4     4

  Columns 21 through 40

    4     4     4     2     2     2     2     2     2     2     2     2     2     2     2
    2     2     2     2     2     2

  Columns 41 through 60
```

```
    2     2     2     2     2     3     3     3     3     3     3     3     3     3
    3     3     3     3     3     3

Columns 61 through 80

    3     4     4     4     4     4     4     4     4     4     4     4     4     4
    4     4     4     4     4     4

Columns 81 through 100

    4     4     4     4     4     4     4     5     5     5     5     5     5     5
    5     5     5     5     5     5

Columns 101 through 120

    5     5     5     5     5     5     5     5     5     5     5     5     5     5
    5     5     5     6     6     6

Columns 121 through 140

    6     6     6     6     6     6     6     6     6     6     7     7     7     7
    7     7     7     7     7     7

Columns 141 through 157

    7     7     7     7     7     7     7     7     7     7     7     7     7     7
    7     7     8
```

codigo/resultado_2.txt

3.3. Código

A continuación incluimos el código que realiza lo anteriormente explicado.

```
% Inicializacion.
close all
clear
clc
colordef white

% Cargo Datos.
load data;

%--------------------------------------------------------------------

%-------------------------
% Construyo matriz total.
%-------------------------
A = hmm4.trans;
B = hmm6.trans;

C = zeros(8, 8);

% Probabilidades de arrancar con una o con otra.
C(1, 2) = 0.5;
C(1, 5) = 0.5;

% Inserto la matriz de HMM4.
C(2:4, 2:4) = A(2:4, 2:4);

% Probabilidades de reiniciar en una o la otra, o terminar.
C(4, 2) = A(4, 5) / 3;
C(4, 5) = A(4, 5) / 3;
C(4, 8) = A(4, 5) / 3;

% Inserto la matriz de HMM6.
C(5:7, 5:7) = B(2:4, 2:4);

% Probabilidades de reiniciar en una o la otra, o terminar.
C(7, 2) = B(4, 5) / 3;
C(7, 5) = B(4, 5) / 3;
C(7, 8) = B(4, 5) / 3;
```

```
% Estado Final.
C(8, 8) = 1;

%--------
% Medias.
%--------
medias = {[], hmm4.means{2}, hmm4.means{3}, hmm4.means{4}, ...
    hmm6.means{2}, hmm6.means{3}, hmm6.means{4}, []};

%------------
% Varianzas.
%------------
varianzas = {[], hmm4.vars{2}, hmm4.vars{3}, hmm4.vars{4}, ...
    hmm6.vars{2}, hmm6.vars{3}, hmm6.vars{4}, []};

%-------------------------------------------------------------------

% Construyo el nuevo modelo.
hmm_total.means = medias;
hmm_total.vars = varianzas;
hmm_total.trans = C;

%-------------------------------------------------------------------

% Genero Secuencia.
[x, q] = genhmm(hmm_total);
[alfas, betas, gamas, xis, proba, p] = calcular_matrices(x, hmm_total);

% Viterbi
[q_opt, proba_opt] = logvit(x, hmm_total);

fprintf('Probabilidad (Log): %f\n', proba);
fprintf('Probabilidad Secuencia Optima (Log): %f\n', proba_opt);

%-------------------------------------------------------------------

% Detector De Tipo De Secuencia.
secuencia_hmm = [];
i = 1;

% Se resuelve con una maquina de estados.
IDLE = 0;
MODELO_4 = 4;
MODELO_6 = 6;

while(q_opt(i) ~= 8)
    switch(q_opt(i))
        case 1
            state = IDLE;
        case 2
            if(state == IDLE)
                state = MODELO_4;
                secuencia_hmm = [secuencia_hmm, MODELO_4];
            elseif(state == MODELO_4)
                state = MODELO_4;
            elseif(state == MODELO_6)
                state = MODELO_4;
                secuencia_hmm = [secuencia_hmm, MODELO_4];
            end
        case 4
            state = IDLE;
        case 5
            if(state == IDLE)
                state = MODELO_6;
                secuencia_hmm = [secuencia_hmm, MODELO_6];
            elseif(state == MODELO_4)
                state = MODELO_6;
                secuencia_hmm = [secuencia_hmm, MODELO_6];
            elseif(state == MODELO_6)
                state = MODELO_6;
            end
        case 7
            state = IDLE;
    end
```

```
114     i = i + 1;
end

% Resultado.
fprintf('La secuencia fue:');
secuencia_hmm
```

codigo/ejercicio_2.m

4. Conclusiones

En este trabajo se pudo trabajar con algoritmos que realizan tanto la generación de secuencias de estados, como sus emisiones. También trabajamos con algoritmos que computan los problemas básicos que pueden plantearse acerca de los modelos ocultos de Markov. El primero de ellos, es obtener la probabilidad de que se de una determinada realización. El segundo, dado que la secuencia de estados de una realización de un modelo oculto de Markov es oculta, poder estimar cual es la mas probable, dadas las emisiones. El tercero, y en el que pusimos mas foco, es en la estimación de los parámetros de un modelo. Esto es importante dado a que es lo que re realiza a la hora de realizar entrenamientos. Y el ultimo de los problemas, dada una realización que puede provenir de combinaciones de modelos, obtener que secuencia de modelos se utilizó para generar esa realización. Todo esto lleva a un mejor entendimiento de la mecánica de los modelos de Markov, de la matemática detrás de los mismos y de las técnicas de computación de los resultados buscados.