Speaker Recognition

INTEGRANTES: Meola Franco Román, Puente Julieta, Strubolini Diego Martín
ITBA Segundo Cuatrimestre 2014

PALABRAS CLAVE: análisis armónico, speaker recognition, mfcc, transformada de fourier, mel-cepstral

Resumen

Este Trabajo Práctico Especial muestra una posible implementación de un reconocedor de habla en Octave basándonos en la transformada de Fourier para el cálculo de los coeficientes mel-cepstrales de distintas muestras de voz.

1. Introducción

El reconocimiento de habla automatizado por una máquina ha sido estudiado por décadas. Hay distintas formas de representaciones paramétricas para señales de acústica. Uno de los más usados es el MFCC o Mel-Frecuency Ceptrum. En el siguiente informe nos basaremos en el paper *An efficient mfcc extraction method in speech recognition* de Wei Han, Cheong-Fat Chan, Chiu-Sing Choy, y Kong-Pang Pun (Ver [1]) y detallaremos una implementación propia en el lenguaje matemático Octave.

2. Implementación

Siguiendo los pasos de [1], detallaremos a continuación las principales funciones implementadas para obtener los coeficientes mel-cepstrales.

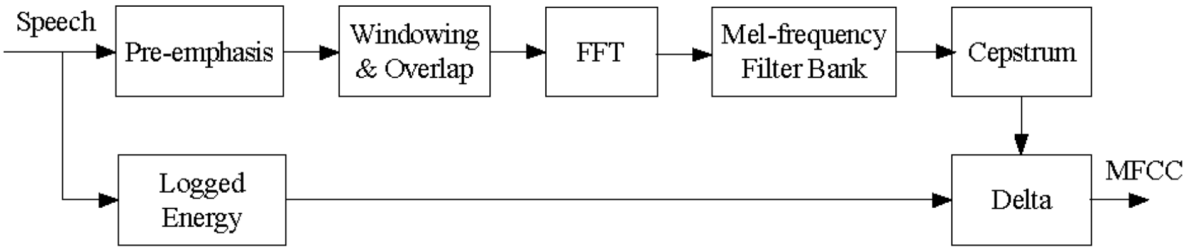

Figura 1: Diagrama de Trabajo

2.1. Pre-emphasis

En esta primera función obtendremos un aplanamiento de la señal. Tanto la constante `a` elegida para este procedimiento (Ver línea 3), como su implementación es similar a la enunciada en la fórmula (1) de [1].

```
1  function yp = preemphasis(y, rows)
2
3  a = 0.97;
4  yp(1) = y(1);
5
6  for n = 2:rows
7      yp(n) = y(n) - a * y(n-1);
8  end
9
10 yp;
```

Código 1: Pre-emphasis

2.2. Windowing and Overlap

Esta etapa nos ayuda a reducir los cambios bruscos del principio y fin de la trama en donde va de cero a la señal y de la señal a cero respectivamente. Para esta función también utilizamos los mismos valores que en [1]. En la misma recibimos el vector de Hamming para ahorrarnos el cálculo del mismo en cada iteración. En la línea 5 utilizamos el vector de Hamming generado por el método `hamming` de Octave.

```
1 function yw = windowing(y,frameSize,frameNmb,overlap,hamming)
2
3 i=1;
4 for k=1 + overlap*(frameNmb -1) : frameSize + overlap*(frameNmb-1)
5     yw(i)=y(k)*hamming(i);
6     i=i+1;
7 end
8
9 yw;
```

Código 2: Windowing

2.3. FFT

Para el cálculo de la transformada de Fourier utilizamos el método `fft` provisto por Octave. Éste método utiliza la Transformada Rápida de Fourier vista en clase.

2.4. Mel-frequency Filter Bank

En esta etapa tomamos como referencia el tutorial sugerido por la cátedra [4]. En la función `filterbanks` de [4] se obtienen 26 filtros, pero en nuestro caso elegimos utilizar 33 filtros como en la descripción de [1]. Por el mismo motivo utilizamos 256 puntos de la transformada de Fourier obtenida en la etapa anterior. Luego de revisar los detalles de [4] decidimos utilizar 300 como el valor del parámetro `min`, ya que usando 0 (como sugiere [1]) no obtuvimos los resultados esperados.

```
1  function fb = filterbanks(min,max,amount,fftsize)
2
3  mmax = mel(max);
4  mmin = mel(min);
5  step = (mmax-mmin)/(amount+1);
6
7  for k=1:amount+2
8      num = (k-1)*step + mmin;
9      f(k) = num;
10     fm(k) = melinv(num);
11     fb(k) = floor((fftsize+1)*fm(k)/(max*2));
12 end
13
14 fbank = zeros(amount,fftsize/2+1);
15
16 for j=1:amount
17     for i=fb(j):fb(j+1)
18         fbank(j,i) = (i - fb(j))/(fb(j+1)-fb(j));
19     end
20     for i=fb(j+1):fb(j+2)
21         fbank(j,i) = (fb(j+2)-i)/(fb(j+2)-fb(j+1));
22     end
23 end
24
25 fb = fbank;
26 fb;
```

Código 3: filterbanks

El ciclo `for` de la línea 16 fue extraído de la implementación en Python de [4].

2.5. Cepstrum

En el siguiente ciclo calculamos los primeros 12 coeficientes mel-cepstrales. En el mismo utilizamos la ecuación (4) de [1].

```
for n = 1 : (coef_amount - 1)
    c = 0;
    for k = 1 : filter_amount
        c += log(filterenergies(k))*cos(n*(k-0.5)*pi/filter_amount);
    end
    coef(n) = c;
end
```

Código 4: Extracto de la función mfcc

2.6. Logged Energy

Para calcular el décimo-tercer coeficiente mel-cepstral utilizamos la función (5) de [1].

```
function energy = loggedenergy(y,framesize)

energy = 0;
for n = 1 : framesize
    energy += y(n)**2;
end;

energy = log(energy);
energy;
```

Código 5: Logged Energy

2.7. Deltas

En esta etapa obtuvimos los otros trece coeficientes (coeficientes deltas) para lograr el total de veintiséis coeficientes `mfcc` necesarios utilizando la función (6) de [1]. Luego de solicitar asistencia a la cátedra, en los casos extremos donde el índice se va de rango se utilizó el vector nulo.

```
delta(1,:) = (2*(mfcc(:,3)) + (mfcc(:,2)))/10;
delta(2,:) = (2*(mfcc(:,4)) + (mfcc(:,3) - mfcc(:,1)))/10;
for f = 3 : (total_frames-2)
    delta(f,:) = (2*(mfcc(:,f+2) - mfcc(:,f-2)) + (mfcc(:,f+1) - mfcc(:,f-1)))/10;
end
delta(total_frames-1,:) = (2*(-1*mfcc(:,total_frames-3)) + (mfcc(:,total_frames) - mfcc
    (:,total_frames-2)))/10;
delta(total_frames,:) = (2*(-1*mfcc(:,total_frames-2)) + (-1*mfcc(:,total_frames-1)))
    /10;
```

Código 6: Extracto de la función mfcc

2.8. MeanDist y VQ

Luego de obtener los coeficientes `mfcc` de cada trama se extrajeron 16 vectores representativos a partir de la función `vq`. Estos vetores se compararon con los `mfcc` de los audios de entrenamiento, utilizando la función `meandist` provista por la cátedra.

3. Resultados

Luego de probar todos los audios de prueba y comparándolos con los audios de prueba obtuvimos los siguientes resultados. En las tablas detallamos:

- Persona: el nombre de la persona que grabó el audio de prueba.

- Coincidencia: Si el programa identificó correctamente el audio de entrenamiento.
- Falsa Coincidencia: En caso de no haber coincidencia se indica el nombre de la persona que el programa asumió.

Se muestran los resultados para las frases de prueba "*Susana*" y "*Juan*".

Persona	**Coincidencia**	**Falsa Concidencia**
Franco	No	Enzo
Sandra	Si	-
Enzo	Si	-
Paula	Si	-
Diego	Si	-
Monica	No	Julieta
Julieta	No	Sandra
Daniela	No	Juli
Agostina	No	Sandra
Sebastián	Si	-

Cuadro 1: Susana: 50 % de efectividad

Persona	**Coincidencia**	**Falsa Concidencia**
Franco	No	Sandra
Sandra	No	Enzo
Enzo	Si	-
Paula	Si	-
Diego	Si	-
Monica	No	Julieta
Julieta	Si	-
Daniela	Si	-
Agostina	No	Julieta
Sebastián	Si	-

Cuadro 2: Juan: 60 % de efectividad

Como se puede ver la efectividad es mayor al 50 %, lo que consideramos un resultando decente pero no ideal. Hay que tener en cuenta que si tomamos como acertada la elección de cualquier miembro de la familia y del mismo sexo para un sujeto de prueba, la tasa de efectividad aumenta a un 80 %. Esto se ve reflejado en las pruebas de Enzo con Franco (Padre e Hijo), Mónica y Julieta (Madre e Hija) y Agostina con Julieta (Hermanas). Asumimos que la frecuencia entre voces de una misma familia, siendo las mismas de mismo sexo, son similares.

4. Problemas Encontrados

Uno de los principales problemas encontrados fue que con ciertos archivos de audio los vectores generados por nuestra implementación variaban considerablemente en dimensión, lo que generaba problemas de rangos inválidos. Una de las características que encontramos en común entre estos archivos eran que contenían silencios absolutos en el medio, lo que suponemos fue la causa de estos problemas.

5. Conclusiones

Como conclusión y para terminar, creemos que deberíamos haber elegido frases de prueba de mayor longitud (no de una sola palabra). Decidimos trabajar con frases cortas para reducir el tiempo de procesamiento de la función, que luego nos dimos cuenta que sacrificó efectividad. Si bien, en un principio esperábamos tasas de efectividad más altas a las conseguidas, los resultados conseguidos siguen siendo mejores a una simple elección aleatoria.

Referencias

[1] Wei Han, Cheong-Fat Chan, Chiu-Sing Choy, and Kong-Pang Pun. An efficient mfcc extraction method in speech recognition. In Proceedings IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006., pa- ges 4 pp.–, May 2006.

[2] Md Rashidul Hasan, Mustafa Jamil, Md Golam Rabbani, and Md Saifur Rahman. Speaker identification using mel frequency cepstral coefficients. In 3rd International Conference on Electrical & Computer Engineering ICECE, volume 2004, 2004.

[3] Y. Linde, A Buzo, and R.M. Gray. An algorithm for vector quantizer design. IEEE Transactions on Communications, 28(1):84–95, Jan 1980.

[4] James Lyons. Mel frequency cepstral coefficient (mfcc) tutorial. `http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`

[5] Manual de Funciones de Octave `https://www.gnu.org/software/octave/doc/interpreter/`

6. Funciones Auxiliares

```
1 function m = mel(f)
2
3 m = 1125*log(1+f/700);
4
5 m;
```

Código 7: Mel

```
1 function m = melinv(f)
2
3 m = 700*(exp(f/1125)-1);
4
5 m;
```

Código 8: Mel Inversa

```
1 function yp = periodogram(y,frameSize)
2
3 for k=1 : frameSize
4   yp(k)= (abs(y(k))**2)/frameSize;
5 end
6
7 yp;
```

Código 9: Periodogram

Índice