

Práctica de Audio

Procesamiento Digital del Audio y Habla

Curso 2020-2021

Sofía Bonifasi Asturias
Sandra Cea Torrescana
Rogelio Sansaloni Sanjuan

ÍNDICE

I.	Estudios.....	3
A.	KNN	3
B.	CART.....	3
C.	SVM.....	4
D.	GMM	5
II.	Resultados y Tablas	7
A.	Tabla de KNN	9
B.	Tabla de CART.....	9
C.	Tabla de SMV	10
D.	Tabla de GMM	10
III.	Matriz de Confusión	11

I. Estudios

A. KNN

Para el KNN, los parámetros que editamos fueron: el Window y el OverlapLength. También agregamos los parámetros de "mfcc", "melSpectrum", "pitch" y "spectralCentroid" con valores de *true*.

```
function aFE = audioFE ()
    %Variable usada para definir los atributos que analizaremos de los audios
    fs = 44100;
    %Se definen los valores de longitud de la trama y numero de
    %muestras solapadas
    aFE = audioFeatureExtractor(...
        "SampleRate",fs, ...
        "Window",hamming(round(0.6*fs),"periodic"), ...
        "OverlapLength",round(0.5*fs), ...
        "mfcc",true, ...
        "melSpectrum",true, ...
        "pitch",true, ...
        "spectralCentroid",true);
end
```

El código en Matlab de la función que implementa KNN es el siguiente:

```
function ypred = KNN(learnDB, learnGT, testDB, k)
%Esta función aplica el clasificador KNN
%KNN devuelve los K valores más cercanos según la distancia euclidiana
knnModel = fitcknn(learnDB, learnGT,'NumNeighbors',k,'Distance','euclidean','Standardize',
ypred = predict(knnModel,testDB);
end
```

Tal y como indica el enunciado, creamos un modelo KNN mediante *fitcknn*, al que le pasamos la distancia k (definida como *knn_K* en *practica_audio*) y usamos *predict* para obtener el *testGT* predicho por el clasificador.

B. CART

Para el CART, los parámetros que editamos fueron: el Window y el OverlapLength. También agregamos los parámetros de "mfcc", "melSpectrum", "pitch" y "spectralCentroid" con valores de *true*.

```
function aFE = audioFE ()
    %Variable usada para definir los atributos que analizaremos de los audios
    fs = 44100;
    %Se definen los valores de longitud de la trama y numero de
    %muestras solapadas
    aFE = audioFeatureExtractor(...
        "SampleRate",fs, ...
        "Window",hamming(round(0.8*fs),"periodic"), ...
        "OverlapLength",round(0.7*fs), ...
        "mfcc",true, ...
        "melSpectrum",true, ...
        "pitch",true, ...
        "spectralCentroid",true);
end
```

El código en Matlab de la función que implementa CART es el siguiente:

```
function ypred = CART(learnDB, learnGT, testDB)
%Esta función aplica el clasificador CART.
%CART utiliza preguntas binarias para determinar en que parte de un mapa de dos
%dimensiones se encuentra cierto valor. Para ello se crea un diagrama en
%forma de regression tree.
treeModel = fitctree(learnDB, learnGT);
ypred = predict(treeModel,testDB);
end
```

Tal y como indica el enunciado, creamos un modelo CART mediante *fitctree* y usamos *predict* para obtener el *testGT* predicho por el clasificador.

C. SVM

Para el SVM, los parámetros que editamos fueron: el Window y el OverlapLength. Además, agregamos los parámetros de "mfcc", "pitch", "spectralSpread", "spectralKurtosis" y "spectralCentroid" con valores de *true*.

```
function aFE = audioFE ()
    %Variable usada para definir los atributos que analizaremos de los audios
    fs = 44100;
    %Se definen los valores de longitud de la trama y numero de
    %muestras solapadas
    aFE = audioFeatureExtractor(...
        "SampleRate",fs, ...
        "Window",hamming(round(0.2*fs),"periodic"), ...
        "OverlapLength",round(0.1*fs), ...
        "mfcc",true, ...
        "pitch",true, ...
        "spectralKurtosis",true, ...
        "spectralSpread",true, ...
        "spectralCentroid",true);
end
```

El código en Matlab de la función que implementa SVM es el siguiente:

```
function ypred = SVM(learnDB, learnGT, testDB)
    %Esta función aplica el clasificador SVM.
    t = templateSVM('KernelFunction','rbf','Standardize',true);
    svmModel = fitcecoc(learnDB, learnGT,'Learner',t);
    ypred = predict(svmModel, testDB);
end
```

Tal y como indica el enunciado, creamos un modelo SVM mediante *fitcecoc*, al que le pasamos el template generado, el cual obtenemos indicándole que queremos usar la función Kernel. Posteriormente usamos *predict* para obtener el *testGT* predicho por el clasificador.

D. GMM

Para el GMM, los parámetros que editamos fueron: el Window y el OverlapLength. Además, agregamos los parámetros de "mfcc", "pitch", "gtcc" y "spectralCentroid" con valores de *true*.

```
function aFE = audioFE ()
    %Variable usada para definir los atributos que analizaremos de los audios
    fs = 44100;
    %Se definen los valores de longitud de la trama y numero de
    %muestras solapadas
    aFE = audioFeatureExtractor(...
        "SampleRate",fs, ...
        "Window",hamming(round(0.6*fs),"periodic"), ...
        "OverlapLength",round(0.5*fs), ...
        "mfcc",true, ...
        "gtcc",true, ...
        "pitch",true, ...
        "spectralCentroid",true);
end
```

El código en Matlab de la función que implementa GMM es el siguiente:

```
function ypred = GMM(learnDB, learnGT, testDB, k)
%Esta función aplica el clasificador GMM.
%Para cada categoría se crea un modelo gmmModel
categories = unique(learnGT);
numCategories = size(categories);
gmmModel = cell(1, length(numCategories));
gmmPDF = zeros((size(testDB,1)), length(numCategories));
ypred = [];
%entrenamos un gmmModel para cada categoría
for i = 1: numCategories
    name_category = categories(i);
    positions = find(learnGT == name_category);
    data_category = [];
    for n = 1:size(positions)
        data_category = [data_category; learnDB(positions(n),:)];
    end
    gmmModel{i} = fitgmdist(data_category,k,'RegularizationValue',0.1,'CovarianceType','diagonal');
end
%Aplicamos cada gmmModel a cada fila de testDB, y obtenemos la maxima
%probabilidad para asignarle la categoría
for n = 1:size(testDB)
    for i = 1:length(gmmModel)
        gmmPDF(n,i) = pdf(gmmModel{i}, testDB(n,:));
    end
    valor_maximo = max(gmmPDF(n,:));
    indice_categoria = find(gmmPDF(n,:)==valor_maximo);
    categoria = categories(indice_categoria,1);
    ypred = [ypred; categoria];
end
```

Esta función es bastante distinta a la que usamos para los demás clasificadores. Inicialmente se generan diversos modelos GMM, uno por cada categoría. La forma de entrenar estos modelos es conseguir generar una matriz de aprendizaje(*data_category*) únicamente con los atributos de aquellas tramas de la categoría que me interese. Para generar el modelo se usa *fitgmdist*, al que le hemos puesto una regularización de 0.1, y una covarianza diagonal para que pudiera converger.

Posteriormente, lo que se hace es generar diversas probabilidades con *pdf* para cada trama de *testDB* y para cada categoría, de forma que se obtiene un vector de 15 columnas (15 categorías) de probabilidad para cada trama de *testDB*. Luego se obtiene la máxima probabilidad (función *max*) para cada trama, es decir, a que categoría es más probable que pertenezca. Se repite el proceso para cada trama de *testDB* y se obtiene el *testGT* predicho.

II. Resultados y Tablas

Para realizar los estudios, hicimos dos pruebas para cada uno de los clasificadores; la primera prueba con el 5% de los datos y la segunda con el 50%. Una vez hechos los estudios, obtuvimos la siguiente tabla con el porcentaje de exactitud para cada método de clasificación de acuerdo al porcentaje de datos utilizado:

Clasificador	Porcentaje de Datos	Porcentaje de Exactitud
KNN	5%	80.60%
	50%	48.21%
CART	5%	83.37%
	50%	54.07%
SMV	5%	78.80%
	50%	49.89%
GMM	5%	70.97%
	50%	42.39%

Como podemos ver, el método más exacto fue el CART. Para realizar cada estudio, primero fue necesario quitar, agregar y editar parámetros con solo el 5% de los datos para que la computadora con la que se hacían las pruebas no se saturara. Una vez los estudios regresaban más de 70% de exactitud, pudimos hacer las pruebas finales con el 50% de los datos. A continuación, mostraremos las tablas que generó Matlab con cada estudio usando el 50% de datos.

Por tanto, el orden la de las *practica_audio* es el siguiente:

- *practica_audio1*: clasificador CART con los parámetros de *audioFeatureExtractor* explicados anteriormente.
- *practica_audio2*: clasificador KNN con los parámetros de *audioFeatureExtractor* explicados anteriormente.
- *practica_audio3*: clasificador SVM con los parámetros de *audioFeatureExtractor* explicados anteriormente.
- *practica_audio4*: clasificador GMM con los parámetros de *audioFeatureExtractor* explicados anteriormente.

También se incluye un *practica_audio* sin ninguna enumeración, que hemos usado con la configuración del mejor accuracy (es decir, *practica_audio1*) para obtener la matriz de confusión.

A continuación, se muestran las matrices obtenidas con cada clasificador:

A. Tabla de KNN

Name ^	Value
accuracy	[46.9717;54.9874;51.2634;39.6070]
accuracy_mean	48.2074
ads	1x1 audioDatastore
ads1	1x1 audioDatastore
ads2	1x1 audioDatastore
features	1x4 cell
i	4
input	1x1 categorical
input2	1x1 categorical
inputArray	1x95 categorical
inputArray2	1x95 categorical
k	4
knn_K	10
koPositions	1x4 cell
labels	1x4 cell
learnDB	148010x47 double
learnGT	148010x1 categorical
methodLM	'knn'
n	517
n2	95
nAudios	517
nTramas	95
okPositions	1x4 cell
path	'./Ficheros/Categorias'
percentOfDataSet	0.5000
testDB	49115x47 double
testGT	49115x1 categorical
testPred	49115x1 categorical

B. Tabla de CART

Name ^	Value
accuracy	[47.6316;59.262...]
accuracy_mean	54.0742
ads	1x1 audioDatast...
ads1	1x1 audioDatast...
ads2	1x1 audioDatast...
features	1x4 cell
Fs	44100
gmm_N	2
i	4
input	1x1 categorical
input2	1x1 categorical
inputArray	1x93 categorical
inputArray2	1x93 categorical
k	4
knn_K	10
koPositions	1x4 cell
labels	1x4 cell
learnDB	160611x47 double
learnGT	160611x1 catego...
methodLM	'cart'
n	575
n2	93
nAudios	575
nTramas	93
okPositions	1x4 cell
path	'/Users/Sofi/Docu...
percentOfData...	0.5000
prompt	'Por favor, escrib...
testDB	53475x47 double
testGT	53475x1 categor...
testPred	53475x1 categor...

C. Tabla de SMV

Name ▲	Value
accuracy	[45.7409;56.868...
accuracy_mean	49.8873
ads	1x1 audioDatast...
ads1	1x1 audioDatast...
ads2	1x1 audioDatast...
features	1x4 cell
Fs	44100
gmm_N	2
i	4
input	1x1 categorical
input2	1x1 categorical
inputArray	1x99 categorical
inputArray2	1x99 categorical
k	4
knn_K	10
koPositions	1x4 cell
labels	1x4 cell
learnDB	170973x17 double
learnGT	170973x1 catego...
methodLM	'svm'
n	575
n2	99
nAudios	575
okPositions	1x4 cell
path	'/Users/Sofi/Docu...
percentOfData...	0.5000
testDB	56925x17 double
testGT	56925x1 categor...
testPred	56925x1 categor...

D. Tabla de GMM

accuracy	[39.4892;45.4784;43.2183;41...
accuracy_mean	42.3875
ads	1x1 audioDatastore
ads1	1x1 audioDatastore
ads2	1x1 audioDatastore
ans	1x1 audioFeatureExtractor
features	1x4 cell
gmm_N	15
gmmPDF	5723x15 double
i	4
input	1x1 categorical
input2	1x1 categorical
inputArray	1x95 categorical
inputArray2	1x95 categorical
k	4
knn_K	10
koPositions	1x4 cell
labels	1x4 cell
learnDB	164065x28 double
learnGT	164065x1 categorical
methodLM	'gmm'
n	575
n2	95
nAudios	575
nTramas	95
okPositions	1x4 cell
path	'./Ficheros/Categorias'
percentOfDat...	0.5000
testDB	54625x28 double
testGT	54625x1 categorical
testPred	54625x1 categorical

III. Matriz de Confusión

La matriz de confusión que se obtiene con el mejor resultado, es decir, con practica_audio1 con el 5% de los datos, es la siguiente:

confusionMatrix															
15x15 double															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	264	0	0	2	6	0	2	26	0	0	0	20	15	5	32
2	1	320	0	0	0	0	0	0	0	4	0	0	0	0	47
3	2	0	323	0	5	0	2	0	0	2	0	0	24	0	14
4	22	0	0	252	0	0	0	0	40	0	0	15	30	0	13
5	0	0	3	0	273	0	0	2	0	16	0	0	71	0	7
6	0	0	0	0	0	248	0	9	0	0	0	0	22	0	0
7	0	0	0	0	0	0	346	0	0	15	0	0	0	0	11
8	0	0	0	1	0	0	0	287	52	0	0	0	32	0	0
9	0	0	0	0	0	3	0	8	335	0	26	0	0	0	0
10	0	7	0	0	34	0	0	0	0	256	0	0	0	0	75
11	0	0	0	0	0	0	0	0	3	0	369	0	0	0	0
12	2	0	0	4	0	13	0	9	2	3	0	277	56	0	6
13	37	5	0	1	36	0	8	23	0	12	0	0	231	0	19
14	11	0	0	0	0	0	0	0	0	0	0	0	0	353	8
15	12	23	4	22	20	0	7	1	0	18	0	0	6	2	257

Los índices de las filas y columnas representan el nombre de las categorías ordenadas alfabéticamente, como se puede observar en la siguiente matriz de categorías:

	1
1	beach
2	bus
3	cafe-restau...
4	car
5	city_center
6	forest_path
7	grocery_st...
8	home
9	library
10	metro_stati...
11	office
12	park
13	residential_...
14	train
15	tram

El código para obtener la matriz de confusión es el siguiente, que se encuentra en el archivo *practica_audio.m*:

```
%Calculo de la matriz de confusión
categories = unique(testGT);
numCategories = size(categories,1);
matrizConfusion = zeros(numCategories);
for i = 1:size(testGT)
    name_category_real = testGT(i);
    name_category_predict = testPred(i);
    position_real = find(categories == name_category_real);
    position_predict = find(categories == name_category_predict);
    matrizConfusion(position_real, position_predict) = matrizConfusion(position_real, position_predict) + 1;
end
```
