CLP Lab 3 Report

Albert Aparicio Isarn

albert.aparicio.isarn@alu-etsetb.upc.edu

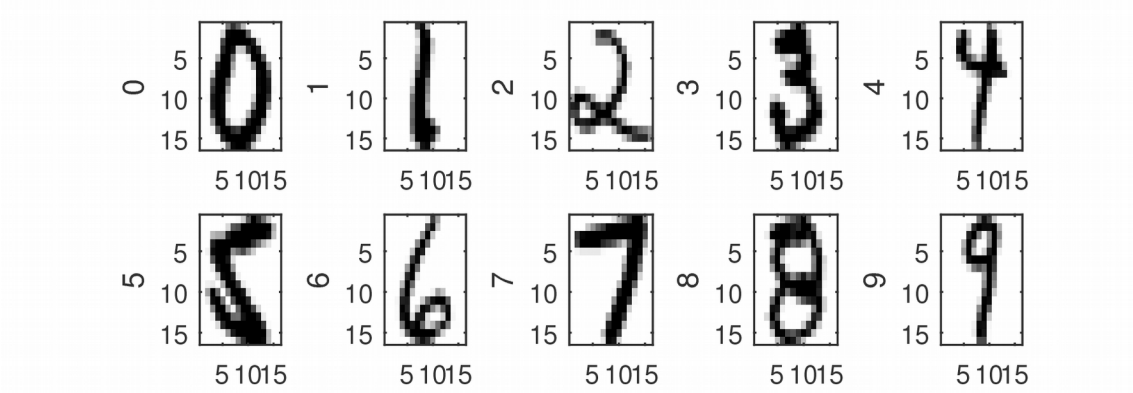
Héctor Esteban

hect.esteban@gmail.com

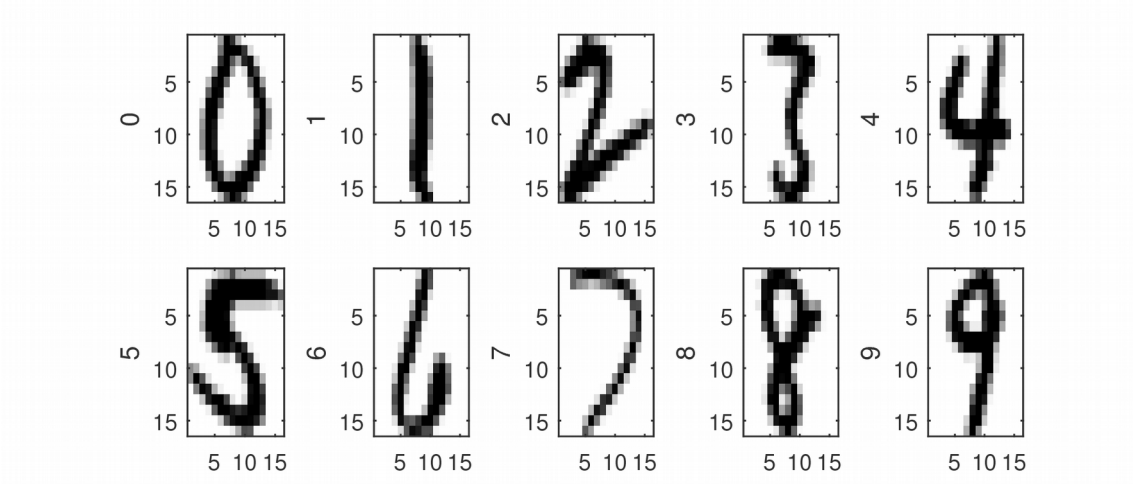
2.3.1. Método KNN

1.

TEST



TRAIN



2.

Se ha elegido la opción de no transformar:

Clasificador\Fase	Training	Test
Linear	0.0608309	0.0969817
KNN	0.0479723	0.0522019

CM_Linear_test =

```
  652    0    1    1    3    2   11    0    5    0
    0  546    0    0    1    1    2    0    0    2
    5    0  340   16   13    2    4    3   19    2
    4    0    5  327    2    8    0    3    7    2
    2   15    7    0  317    1    6    0    6   16
    7    1    4   18    7  257    3    0   11    3
    4    2    3    0    9    2  340    0    3    0
    0    1    0    3    7    0    0  289    2   42
    5    2    4   14   13    9    3    1  252    5
    1    1    0    1   12    0    0   10    2  330
```

CM_knn_test =

```
  670    0    1    1    1    1    1    0    0    0
    0  549    0    0    1    0    2    0    0    0
    8    4  371    1    3    0    1   10    5    1
    3    0    2  342    0    4    0    2    2    3
    1   16    5    0  328    0    3    1    1   15
    8    1    1   10    1  279    6    0    0    5
    3    1    2    0    3    0  354    0    0    0
    0    4    2    0    4    0    0  323    1   10
    2   10    1   10    4    3    1    2  272    3
    2    0    0    1    2    0    0    9    0  343
```

El valor elegido es **k** = 10.

Visto los resultados de estas matrices podemos ver que:

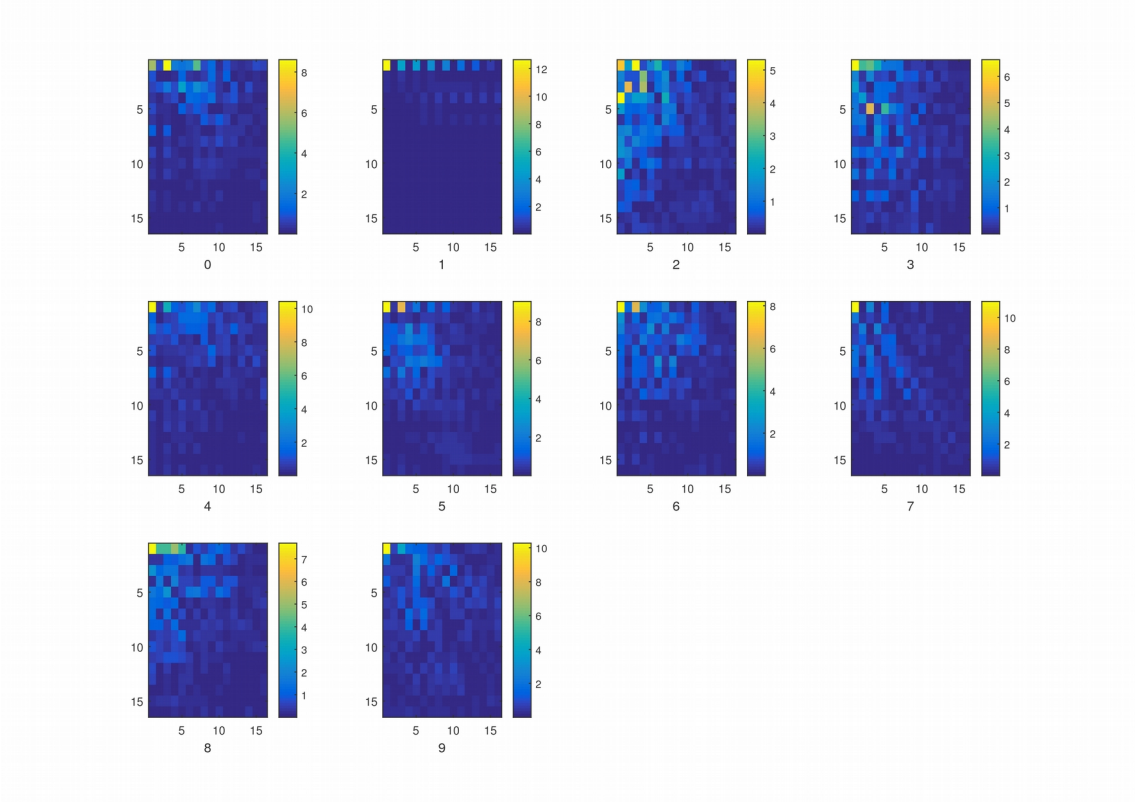
- Para el clasificador lineal la mayor tasa de error se produce cuando tenemos que clasificar un 7 (muchas veces se decidirá por clasificar como si fuera un 9) y cuando tenemos que hacerlo para un 2 (a veces decide 3 y 8).
- Para el clasificador KNN el problema viene con el 4 (a veces decide un 1 y también un 9)

3.

Principalmente, la energía de las transformaciones DCT se concentra en la componente continua (0,0). Dependiendo de cada carácter, se encuentra más o menos energía en el resto de componentes.

Un caso particular es el del dígito 1. Al estar formado en su práctica totalidad por líneas verticales, la energía se concentra en la fila superior de la transformada. Esta fila corresponde a las funciones base con líneas verticales. Los valores por los que la energía está más repartida parecen ser el 2 y el 3.

Una aplicación de la DCT seria la de eliminar las muestras de alta frecuencia. Estas muestras de la DCT contienen poca energía, de manera que se pierde poca información.



4.

Transformación DCT

Dim = 8

Clasificador\Fase	Training	Test
Linear	0.080366	0.087333
KNN	0.0390702	0.0470064

CM_Linear_test =

```
  658    0    1    5    1    1    4    0    5    0
    0  547    0    1    1    0    1    0    2    0
    3    3  343    6   12    2    8    2   22    3
    2    0    4  321    0   11    0    6   12    2
    4   11    8    0  316    0    3    2    5   21
    9    0    3   16    7  265    3    0    5    3
    1    2    5    0    4    4  344    0    3    0
    0    2    1    0    6    0    0  300    3   32
    4    7    3   16    7    3    1    0  263    4
    0    1    0    1   13    0    0    8    2  332
```

CM_knn_test =

```
  672    0    1    2    0    0    0    0    0    0
    0  550    0    0    1    0    1    0    0    0
   11    3  372    3    1    0    2    5    7    0
    2    1    2  342    0    1    0    1    5    4
    1    8    3    0  334    0    4    2    1   17
    8    0    2    6    0  287    3    0    3    2
    7    2    1    0    1    1  350    0    1    0
    0    4    1    0    4    0    0  330    0    5
    3    6    0    7    4    1    2    3  277    5
    1    1    1    1    7    0    0    6    2  338
```

Dim = 4

Clasificador\Fase	Training	Test
Linear	0.162216	0.157595
KNN	0.0699802	0.0809005

CM_Linear_test =

```
  599   39    3    4    4    1    9    0   16    0
    0  546    0    0    1    1    1    0    2    1
    4   29  309   18   18    2    5    0   16    3
    2   15    2  302    1   14    0   12    3    7
    0   42    6    0  304    0    1    2    1   14
    4    5    1   36    7  238    4    0   13    3
    4   18    3    0    2    1  333    0    2    0
    0   18    0    2    3    0    0  274    2   45
    5   87    3    5   10    4    1    0  189    4
    0   22    0    1   11    1    0   10    1  311
```

CM_knn_test =

```
  665    4    2    2    0    0    1    0    1    0
    0  547    0    2    2    0    0    0    1    0
    8    2  362    7    5    0    5    3   11    1
    6    1    0  328    1   11    0    4    4    3
    0   11    9    0  339    0    2    1    0    8
   15    0    1   33    3  245    6    0    5    3
    6    6    2    0    1    0  348    0    0    0
    0    1    0    1    7    0    0  318    4   13
   19   40    2    9    4    1    1    0  230    2
    0    3    0    1    7    0    0    9    4  333
```

Dim = 2

Clasificador\Fase	Training	Test
Linear	0.376113	0.377041
KNN	0.313304	0.37234

CM_Linear_test =

```
  498    54    39    30    16     0     3     1    34     0
    6   525     4     1    13     0     2     0     1     0
   70    55   170    25     3     2    69     0     8     2
   29    32     2   187    44    34     0    10     2    18
   11   118     9    40    98    13     4     2    39    36
   29    19     1    40    30   166     0     0    24     2
   12    27    45     0     0     3   276     0     0     0
    0    26     0     1     2     0     0   229     1    85
   63    59     7    11    46     3     0     0   105    14
    1    31     0     5    22     0     0    30     4   264
```

CM_knn_test =

```
  538    27    30    24     6     6    12     0    32     0
   11   513     7     1    11     1     3     0     4     1
  104    23   159    19     7     9    69     0    13     1
   36     9    10   225    34    17     1     5    14     7
   24    48     8    56   108    11     6     6    63    40
   58    12     7    69    22    77     1     0    59     6
   14     2    31     0     2     2   312     0     0     0
    0     0     0     4     9     0     0   249     2    80
   90    34    13    16    41     7     1     0    95    11
    1     4     0     7    24     0     0    54     6   261
```

Transformación Hadamard

Dim = 8

Clasificador\Fase	Training	Test
Linear	0.0776459	0.08857
KNN	0.0430267	0.0566551

CM_Linear_test =

```
  653    2    0    2    2    1    9    0    5    1
    0  544    1    1    2    0    1    0    1    2
    5    2  345   10   18    4    5    4    9    2
    2    2    2  329    0   13    0    4    4    2
    2    7    9    0  317    2    2    1    4   26
   10    2    1   11    9  268    4    0    5    1
    4    2    6    0    5    2  337    0    7    0
    1    3    0    1    5    0    0  306    1   27
    2   16    0   10    7    6    1    2  258    6
    0    3    0    1   12    0    0   13    1  327
```

CM_knn_test =

```
  665    1    4    1    0    1    2    0    0    1
    0  549    1    0    2    0    0    0    0    0
    8    8  365    2    4    0    3    8    6    0
    0    3    3  342    0    5    0    2    1    2
    1    7    3    0  335    0    3    1    0   20
   11    1    2    9    2  275    8    0    1    2
    6    1    1    0    5    0  350    0    0    0
    0    6    0    0    3    0    0  326    1    8
    3   11    1    8    3    2    2    3  273    2
    0    7    0    1    8    0    0    6    2  333
```

Dim = 4

Clasificador\Fase	Training	Test
Linear	0.181009	0.18951
KNN	0.0870425	0.114547

CM_Linear_test =

```
  580   31    1    2    4    1    7    0   49    0
    0  520    0    0    4    2    2    0   22    2
    9   44  305    9   14    2    2    2   17    0
    0   35    5  272    1   19    1    8    5   12
    1   57    1    0  261    0    3    0    4   43
    6   17    0   41    5  226    6    0    7    3
    4   15    1    0    2    2  336    0    3    0
    0   14    0    0    3    0    0  271    2   54
    8   76    6    8    3    5    2    1  194    5
    0   20    0    0   15    0    0   10    1  311
```

CM_knn_test =

```
  649   14    2    2    1    0    1    0    6    0
    0  541    0    1    2    1    3    2    2    0
    9   12  362    9    2    1    0    1    7    1
    6    7   11  298    0   15    0    6    9    6
    2   21    5    0  299    0    2    0    2   39
   17    7    0   26    4  242    8    1    5    1
    6    5    1    0    4    2  344    0    1    0
    0    6    2    0    3    0    0  310    2   21
   20   64    1    9    2    4    2    0  205    1
    1    5    0    0    8    0    0    9    5  329
```

Dim = 2

Clasificador\Fase	Training	Test
Linear	0.347676	0.33523
KNN	0.256182	0.304057

CM_Linear_test =

```
  544   50   16   18    9    3    4    2   29    0
    6  527    5    2    3    0    2    1    6    0
   36   46  255    8    9    2   40    0    8    0
   23   24    4  193   51   13    0   26    5   19
    8   74   20   33   93   51    6    9   12   64
   34   14    3   51   26  162    6    0    5   10
    6   23   38    1    0    1  294    0    0    0
    1   28    0    3    4    0    0  241    4   63
   55   79   14   15   28    4    0    3  107    3
    1   15    0   12   15    2    0   32    9  271
```

CM_knn_test =

```
  580   24   13   13    2    2    3    0   38    0
   17  511    8    2    1    0    0    2   11    0
   51   21  265    4   14    2   33    0   14    0
   36   11    3  226   24   13    0   14   17   14
   15   38   25   53  113   33    6   19   14   54
   41    8    7   37   38  157    4    0   13    6
   11    0   31    0    3    1  317    0    0    0
    2    7    0   13    8    0    0  245    1   68
   74   58    8   25    8    4    1    0  125    5
    1    1    0   20   14    2    0   33   12  274
```

La transformación DCT con 4x4 pixeles es la más eficiente. Su error para la base de datos de test es 0.0809005 . Otras Combinaciones también resultan en error en test inferior al 10% como són DCT dimensión 8 y Hadamard dimensión 8 pero són menos eficientes computacionalmente.

5.

Para k=1, el sistema está sobre-entrenado, ya que asigna la clase del vector más cercano. Como se está trabajando con los datos de training, las clases ya son sabidas.

En training, los vectores que entran al sistema son los de training, por tanto, con k=1, el vecino más cercano a cada vector es él mismo, por tanto, la clase asignada siempre es la del vector mismo.

El código usado para este apartado es el siguiente:

knn_script.m

```
%% Compute KNN for k=1:10
errors = zeros(10,2,10);

for i=1:10
    [errors(:,:,i)] = compute_knn(1,4);
end

mean_errors = mean(errors, 3);

%% Plot results
plot(1:10, mean_errors(:,1), 1:10, mean_errors(:,2))
hold on
grid on
legend('Train Error', 'Test Error', 'Location', 'best');
hold off
```

compute_knn.m

```
%% Create a knn classifier:
aux_errors = zeros(10,2);
for K_neig=1:10
    knnclass = fitcknn(X_train,Labels_train,'NumNeighbors',K_neig);
    knn_out = predict(knnclass,X_train);
    aux_errors(K_neig,1)=sum(Labels_train ~=
knn_out)/length(Labels_train);
%     fprintf(1,' error knn train = %g   \n', aux_errors(K_neig,1))
    knn_out = predict(knnclass,X_test);
    aux_errors(K_neig,2)=sum(Labels_test ~=
knn_out)/length(Labels_test);
%     fprintf(1,' error knn test = %g   \n', aux_errors(K_neig,2))
    % Test confusion matrix
    if i_CM==1
        CM_knn_test=confusionmat(Labels_test,knn_out);
    end
%     aux_errors(i,1), aux_errors(i,2)

%     [knn_Pe] = mean(aux_errors, 1);
end

knn_Pe = aux_errors; % Matrix to be returned
```

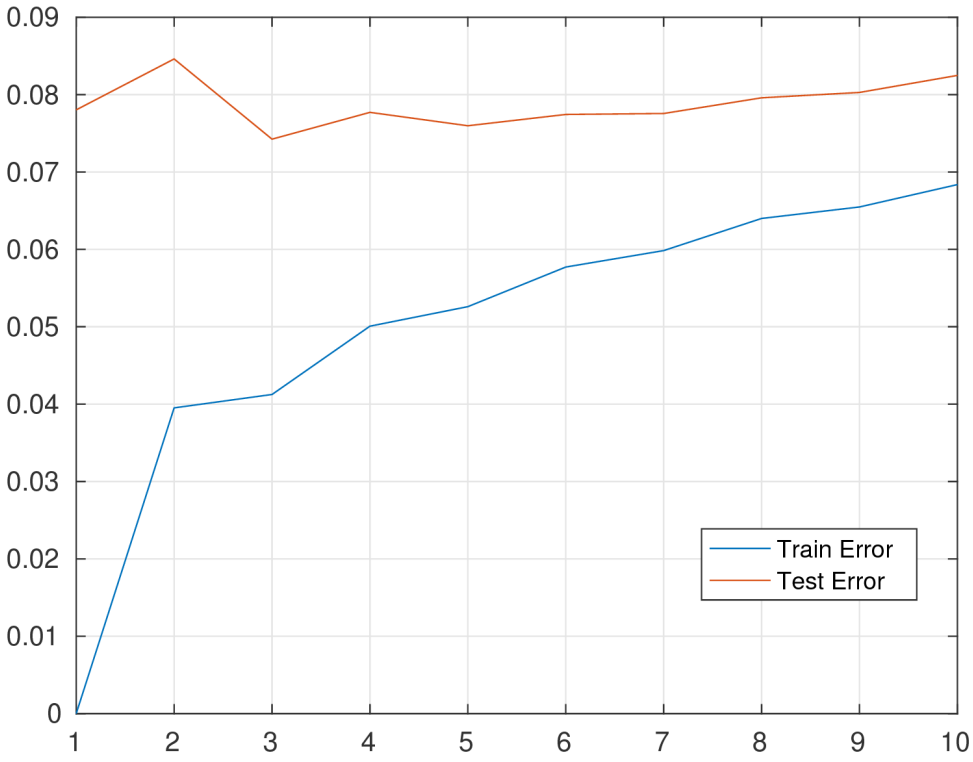


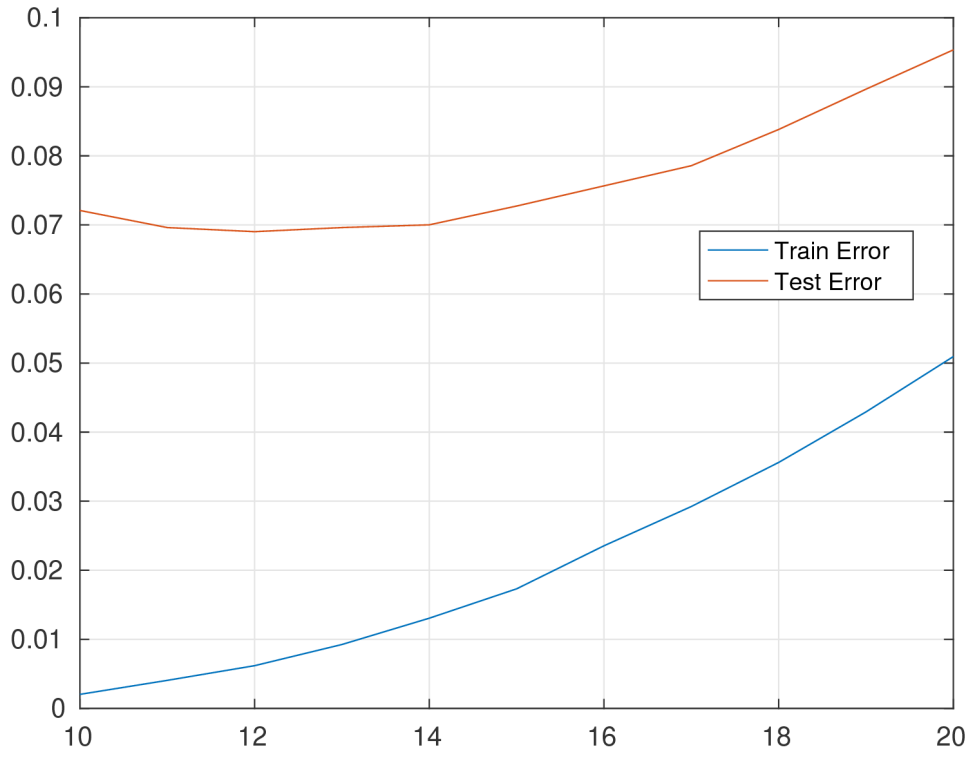
Figura 1. Errores medios de test y training para k=1:10

2.3.2. Método de Parzen

A continuación se muestra la tabla de errores con el método Parzen:

Fase \ h	1	10	20	100
Training	0	0.0032	0.0462	0.5406
Test	0.4805	0.0727	0.0943	0.5532

Habiendo realizado la simulación con el valor de h entre 10 y 20, inclusive, repitiendo 5 veces cada simulación y promediando las probabilidades de error, el gráfico es el siguiente:



El código usado es:

avg_parzen.m

```
%% Compute Parzen error probabilities 5 times and average
errors = zeros(11,2,5);

parfor i=1:5
    i

    [errors(:,:,i)] = prac4_zip_parzen;
end

mean_errors = mean(errors, 3);

%% Plot results
plot(10:20, mean_errors(:,1), 10:20, mean_errors(:,2)), hold on
grid on
legend('Train Error', 'Test Error', 'Location', 'best');

hold off

save('parzen_10_20_5reps_data.mat', 'errors')
```

prac4_zip_parzen.m

```
%% Create a Parzen classifier
% h = [1, 10, 20, 100];
h = 10:20;

Predict_train = zeros(length(X_train), length(h));
Predict_test = zeros(length(X_test), length(h));
for i = 1:length(h)
    Predict_train(:, i) =
predict_parzen(X_train,Labels_train,N_classes,h(i),X_train);
    Predict_test(:, i) =
predict_parzen(X_train,Labels_train,N_classes,h(i),X_test);
end

%% Replicate Labels
Labels_test = repmat(Labels_test, 1, length(h));
Labels_train = repmat(Labels_train, 1, length(h));

%% Compute error probability
% Transpose to get a column array
Train_prob = (1 -
(sum(Labels_train==Predict_train)/length(Labels_train)));
Test_prob = (1 -
(sum(Labels_test==Predict_test)/length(Labels_test)));

Iteration_prob = [Train_prob', Test_prob']; % Matrix to be returned
```

3.2. MicroArray

A continuación se presenta la tabla de probabilidades de error para k=1:4.

Fase \ k	1	2	3	4
Training	0	0.125	0.208333	0.291667
Test	0.36	0.28	0.56	0.44

Como hemos justificado con anterioridad en training cuando utilizamos k=1 al compararse consigo mismo el error de clasificación nos da 0. Las clases seleccionadas para trabajar han sido la 1, 2, 3, 4, 6, 7, 9 y 12. Analizando estas clases los mejores resultados se producen para k=2 donde el error de test es 0.28. Las clases que producen más error de clasificación para k=2 son la clase 3 y clase 4

Las matrices de confusión para los cuatro valores de k son las siguientes:

k = 1
CM_knn_train =

2	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0
0	0	3	0	0	0	0	0
0	0	0	4	0	0	0	0
0	0	0	0	2	0	0	0
0	0	0	0	0	4	0	0
0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	3

CM_knn_test =

2	0	0	0	0	0	0	0
0	3	1	0	0	0	0	0
0	0	1	0	1	1	0	0
0	2	0	1	0	1	0	0
0	1	0	0	2	0	0	0
0	0	0	0	0	3	0	0
0	0	0	0	0	0	2	1
0	0	0	0	1	0	0	2

k = 2

CM_knn_train =

2	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0
1	0	2	0	0	0	0	0
0	1	0	3	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	4	0	0
0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	3

CM_knn_test =

2	0	0	0	0	0	0	0
1	2	0	1	0	0	0	0
0	0	1	1	0	1	0	0
0	2	0	1	0	0	0	1
0	0	0	0	3	0	0	0
0	0	0	0	0	3	0	0
0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	3

k=3

CM_knn_train =

2	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0
1	0	2	0	0	0	0	0
1	1	0	2	0	0	0	0
0	1	0	0	1	0	0	0
0	0	0	0	0	4	0	0
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	3

CM_knn_test =

2	0	0	0	0	0	0	0
1	3	0	0	0	0	0	0
1	0	0	0	0	2	0	0
0	3	0	0	0	0	0	1
0	2	0	0	1	0	0	0
0	1	0	0	0	2	0	0
0	0	0	0	1	0	0	2
0	0	0	0	0	0	0	3

k=4

CM_knn_train =

1	0	0	1	0	0	0	0
0	4	0	0	0	0	0	0
0	0	2	0	0	0	0	1
0	3	0	1	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	4	0	0
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	3

CM_knn_test =

1	0	0	1	0	0	0	0
0	3	0	1	0	0	0	0
1	1	0	0	0	0	0	1
0	0	0	4	0	0	0	0
0	1	0	2	0	0	0	0
0	1	1	0	0	1	0	0
0	0	0	0	0	0	2	1
0	0	0	0	0	0	0	3

4. Base de datos de usuarios de Amazon

Para esta parte de la práctica se ha descargado la base de datos "Amazon Commerce reviews set", disponible en la página siguiente:

http://archive.ics.uci.edu/ml/datasets/Amazon+Commerce+reviews+set

Esta base de datos está formada por reseñas de usuarios de Amazon. Hay 50 usuarios, identificados por un nombre único. De cada usuario hay datos de 30 de sus críticas.

Este dataset tiene sus datos organizados por palabras (se consideran 10000, entre palabras individuales y combinaciones de las mismas), y cada reseña se muestra como una lista de cuantas veces se ha usado cada una de las 10000 palabras posibles.

Cada reseña se localiza en una de las 1500 filas de la matriz de datos, cuya última columna contiene el nombre del usuario que ha escrito la crítica.

El hecho que este nombre esté escrito con letras y no tenga una correspondencia numérica dificulta el problema de la lectura de la base de datos.

Para solucionar este problema, en vez de usar la función `textread`, se usa la función `textscan`, que retorna un `cell array`, que puede contener cualquier tipo de datos. Una vez cargado en memoria, este `cell array` se puede convertir a una matriz corriente, transformando los nombres en números.

El código para leer el archivo de texto y convertirlo en una matriz usable se encuentra en la página siguiente.

El resultado es una matriz de tamaño 1500x10001 (1500=50 usuarios x 30 reseñas; 10001 = 10000 características + 1 nombre) con valores numéricos. Los usuarios se han convertido en números del 0 al 49.

```
% Parse Amazon users dataset
clear

%% Open file
am = fopen('Amazon_initial_50_30_10000.arff', 'r');

%% Read data into cell array
% The headerlines argument skips all lines that are not useful data
data = textscan(am,'%s','headerlines',10005);

% Close file
fclose(am);

% The original 'data' is a 1x1 cell with a 1500x1 cell array inside
data = data{1,1};

%% Parse data into useful matrix
% Split by commas
data_split = split(data, ','); % Result is 1500x10001 string array
data_size = size(data_split);


%% Convert user names into numbers
names = {'Agresti','Ashbacher','Auken','Blankenship','Brody','Brown',...
    'Bukowsky','CFH','Calvinnme','Chachra','Chandler','Chell',...
    'Cholette','Comdet','Corn','Cutey','Davisson','Dent','Engineer',...
    'Goonan','Grove','Harp','Hayes','Janson','Johnson','Koenig',...
    'Kolln','Lawyeraau','Lee','Lovitt','Mahlers2nd','Mark','McKee',...
    'Merritt','Messick','Mitchell','Morrison','Neal','Nigam',...
    'Peterson','Power','Riley','Robert','Shea','Sherwin','Taylor',...
    'Vernon','Vision','Walters','Wilson'};

for i=1:data_size(1)
    % Find the index in 'names' of the name at the end of the i-th vector
    [a,b]=find(data_split(i, end) == names); %b is the name's index

    % Assign the name's index into data_split instead of the name's string
    data_split(i, end) = b-1; % Index names starting from 0
end

%% Convert string array into numeric matrix
data_mat = str2double(data_split);
```