UNIVERSITAT POLITÈCNICA DE CATALUNYA

Sessió 8

INFORME

VISIÓ PER COMPUTADOR

Autors

DAVID LATORRE
ADRIÀ AUMATELL

Introducció

En aquesta sessió treballarem el reconeixement dels caràcters de matrícules de cotxes espanyoles, suposant que la detecció la localització dels caràcters a la imatge ja la sabem.

Aquest reconeixement el farem amb el mètode de classificació d'arbres de decisió. Per tant, per començar, necessitem almenys una imatge principal, la qual contingui tots els caràcters que hi pot haver en una matrícula espanyola, per tal de dotar d'un mínim coneixement de tots els caràcters al nostre model. Per assolir això, utilitzarem aquesta imatge (on es mostren tots els caràcters de forma clara):

Com que l'objectiu d'aquesta sessió no és detectar els caràcters a la imatge sinó reconèixer quins són, aquesta imatge ens va molt bé, ja que un cop l'hem binaritzat podem detectar cada caràcter individualment perquè cada un és una unitat connexa de la imatge.

Codi:

```
i = rgb2gray(imread('joc_de_caracters.jpg'));
figure
imshow(i);

IB = i < 128;
figure
imshow(IB);

CC = bwconncomp(IB);
```

Fet això, ara bé la part més complicada d'aquest projecte, i la que definirà com de bo serà el nostre classificador, ja que ara hem d'extraure i triar amb quines

propietats descriptores dotarem l'arbre de decisions perquè aquest diferencïi entre diversos caràcters.

Per fer un bon algorisme que reconeixi amb una precisió notablement bona els caràcters de les matrícules, necessitarem fer un anàlisi profund d'aquestes propietats, i, segurament, construir-les nosaltres mateixos.

Però per falta de temps, i perquè aquest es tracta d'un petit projecte que té com a objectiu la construcció d'un simple algorisme de reconeixement de caràcters, utilitzarem només característiques de la funció "regionprops" de MATLAB, les quals ens les dona aquesta funció automàticament per a cada unitat connexa de la imatge.

Mirem doncs quines d'aquestes característiques ens interessen més, i quines no hauríem d'utilitzar. El principal aspecte que hem de tenir en compte davant la tria, és que no volem triar propietats que siguin variants de la mida de l'objecte que s'utilitza, ja que a l'hora de detectar una matrícula o una altre les lletres podrien aparèixer més o menys grosses, la mida de la imatge podria ser més o menys grossa... I, en cas que ens interessi alguna característica però aquesta depèn de la mida, llavors l'hem de dividir per alguna variable que faci que ja no sigui depenent de la mida (per exemple dividir l'àrea pel perímetre al quadrat).

Les característiques que hem acabat triant, i que per tant són amb les que dotarem principalment al nostre classificador són (les que comencen amb C):

```
prop = regionprops('table', CC, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'FilledArea', 'MaxFeretProperties', 'MinFeretProperties', 'Solidity');

C1 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C2 = prop.Perimeter ./ prop.BoundingBox(:,3);
C3 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C4 = prop.FilledArea ./ prop.Area;
C5 = prop.EulerNumber;
C6 = prop.Eccentricity;
C7 = prop.Solidity;
C8 = prop.Extent;
C9 = prop.MaxFeretAngle;
C10 = prop.MinFeretAngle;
```

Com es pot observar, hem intentat triar tots aquells descriptors de forma que hem trobat, ja que considerem que aquests seràn els més útils per diferenciar entre els caràcters.

Més endavant comprovarem com aquestes rendeixen davant del reconeixement d'algunes imatges o altres. Ara, però, acabem de construir el nostre algorisme.

Un cop tenim les característiques, les agrupem en forma de vector per tal de passar-les a "TreeBagger" de forma correcta, i també construim un vector que li dirà a "TreeBagger" quina és l'etiqueta de cada caràcter amb la qual ens de referir segons sigui un o altre:

```
X = [C1 C2 C3 C4 C5 C6 C7 C8 C9 C10];
Y = ['0'; '1'; '2'; '3'; '4'; '5'; '6'; '7'; '8'; '9'; 'B'; 'C'; 'D'; 'F'; 'G'; 'H'; 'J'; 'K'; 'L'; 'M'; 'N'; 'P'; 'R'; 'S'; 'T'; 'V'; 'W'; 'X'; 'Y'; 'Z'];
```

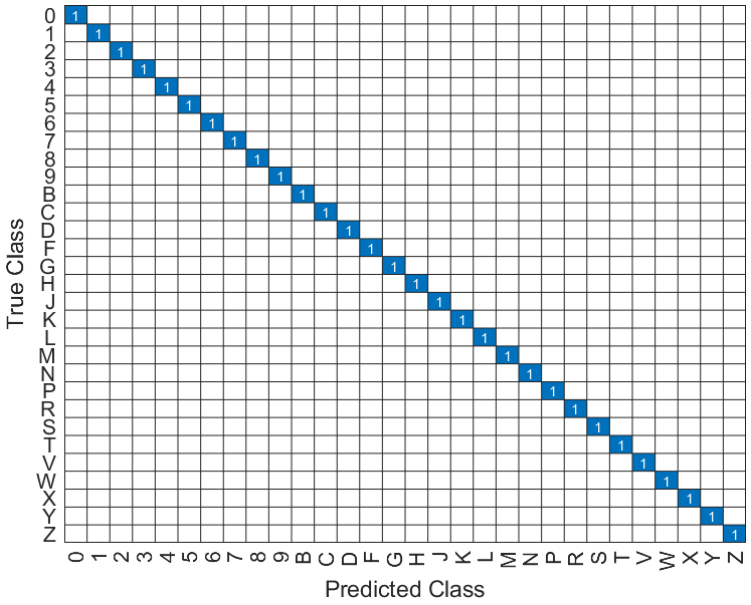
Fet això, ja podem construir els arbres de decisions. En aquest cas considerem suficient amb fer 100 arbre de decisions, no ens cal fer-ne més.

```
Mdl = TreeBagger(100, X, Y); %classificador
```

Ara que ja tenim fet el classificador, assegurem-nos que hem fet tots els passos anteriors correctament comprovant que si apliquem una predicció amb els caràcters de la imatge amb els quals el classificador ha sigut entrenat, aquesta predicció és perfecte: (per mirar si es perfecte fem el "confusionchart" amb els valors que realment fan referència a la imatge)

```
[label, score] = predict(Mdl, X);
N = cell2mat(Mdl.ClassNames);
L = cell2mat(label);
sum(N == L)  %Quantes son igual
figure
confusionchart(N, L);
```

Output:



Veiem que efectivament és compleix això.

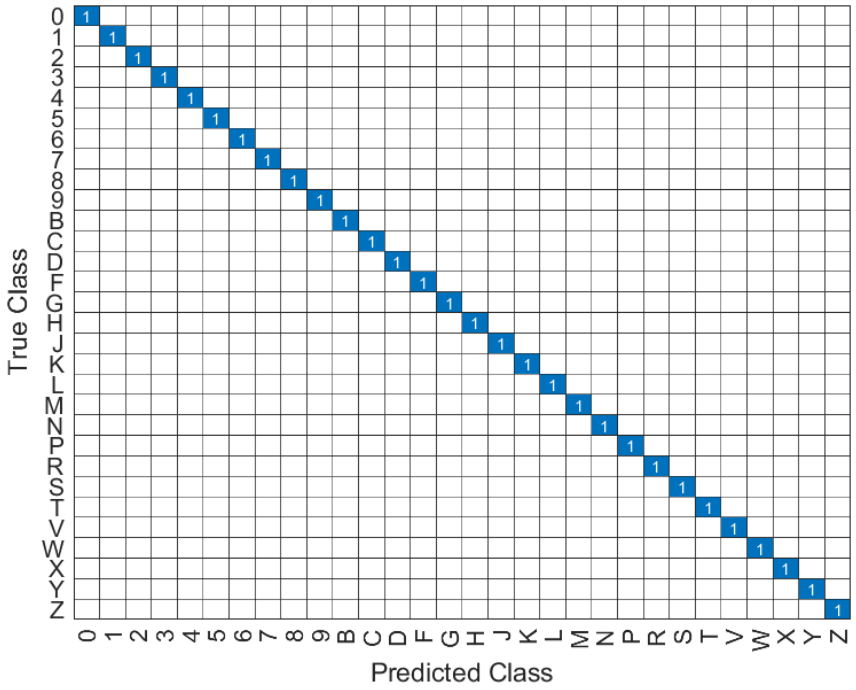
Ja tenim llavors l'algorisme fet. És hora, ara de veure com de bo és aquest observant la qualitat de les prediccions amb altres imatges. És clar que es tracta d'un classificador molt senzill, i que per tant davant de casos on els caràcters no s'assemblin molt respecte a com es troben amb la imatge amb que l'hem entrenat llavors els resultats seran molt dolents, però intentem veure almenys si podem obtenir bons resultats davant d'imatges que s'assemblen.

Apartat a: Imatge x2 i x8

Primer, anem a observar com es comporta l'algorisme amb la imatge amb zoom proporcionada. Per tal d'obtenir la imatge ampliada, senzillament apliquem un imresize a la original.

```
i = rgb2gray(imread('joc_de_caracters.jpg'));
i = imresize(i, 2);
```

Output:

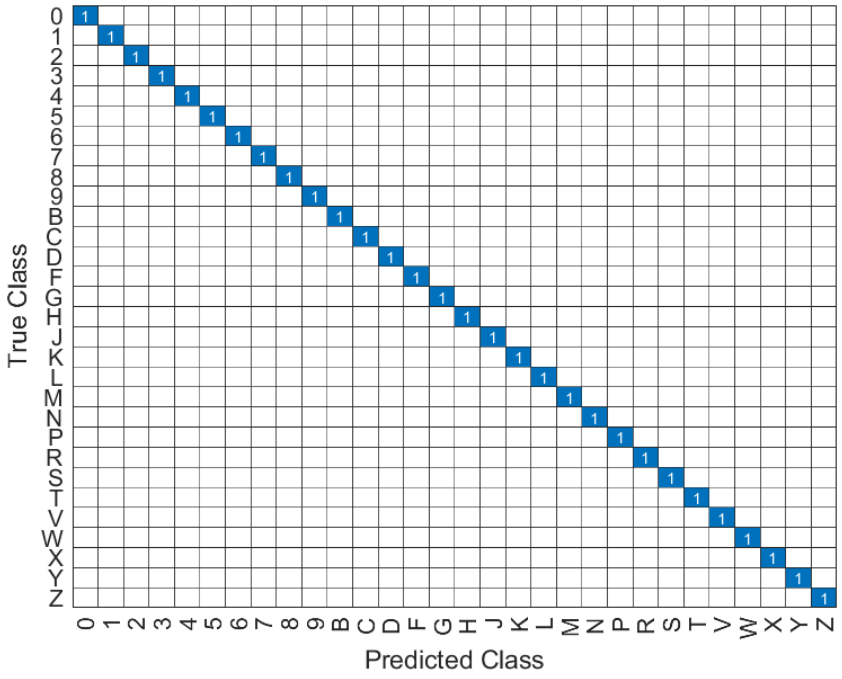


Com es pot observar, els descriptors escollits son prou robustos com per aconseguir detectar correctament tots els caràcters.

Anem a comprovar, també, els resultats si augmentem la mida de la imatge 8 vegades.

Output:

En aquest cas, el resultat

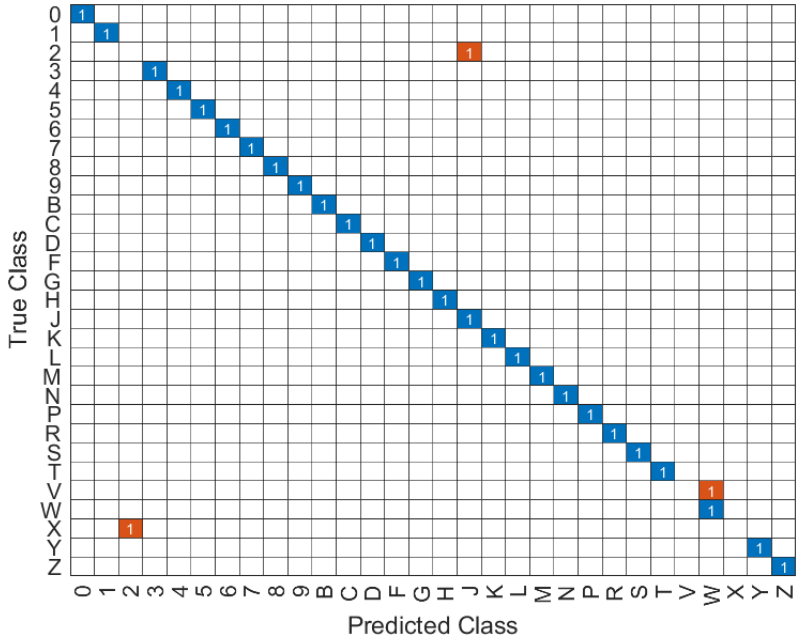


també és del tot correcte, així que podem concloure que els descriptors escollits son suficientment robustos al zoom.

Apartat b: Imatges deformades

En aquesta secció comprovarem la robustesa de l'algorisme amb deformacions dels caràcters. Primer, el posarem a prova amb la imatge amb deformacions lleus.

Output:



Els resultats obtinguts son que l'algorisme ha encertat 27 dels 30 caràcters cosa que, lluny de ser perfecte, representa un 90% d'encerts. Cal tenir en compte que els descriptors s'han escollit per ser robusos al zoom i no específicament a les deformacions. Si tenim present aquesta premisa, el resultat encara és prou accepatble.

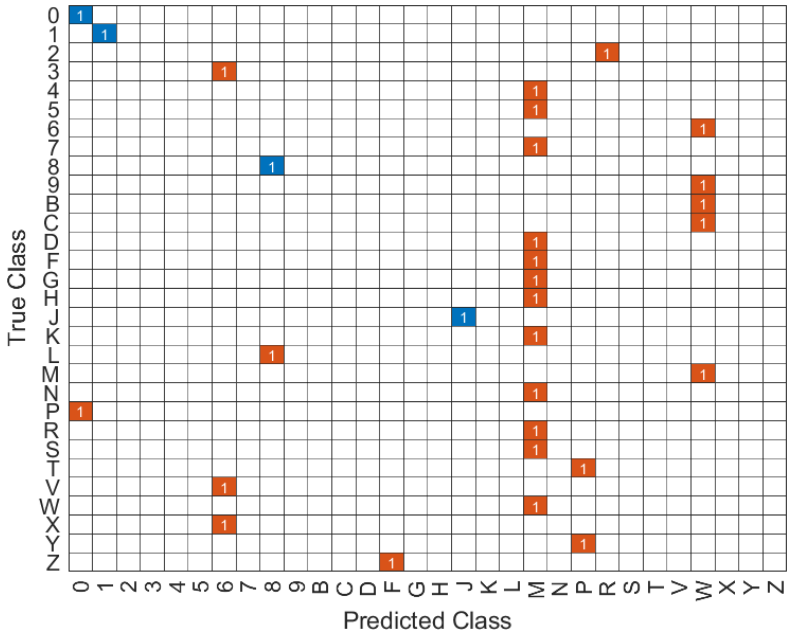
Anem a complicar una mica més les coses testejant l'algorisme amb la imatge amb deformacions importants que podem trobar a la carpeta d'imatges de l'assignatura.

Per poder processar aquesta imatge, primer haurem de realitzar un open perquè, en cas contrari, es detecten més de 30 caràcters.

```
i = rgb2gray(imread('Joc_de_caracters_deformats II.png'));
figure
imshow(i);
IB = i < 128;
figure
imshow(IB);
%IMOPEN NECESSARI SINO DETECTA MES DE 30 OBJECTES
a = strel('disk', 5);
```

Ara sí, ja podem executar l'algorisme.

Output:



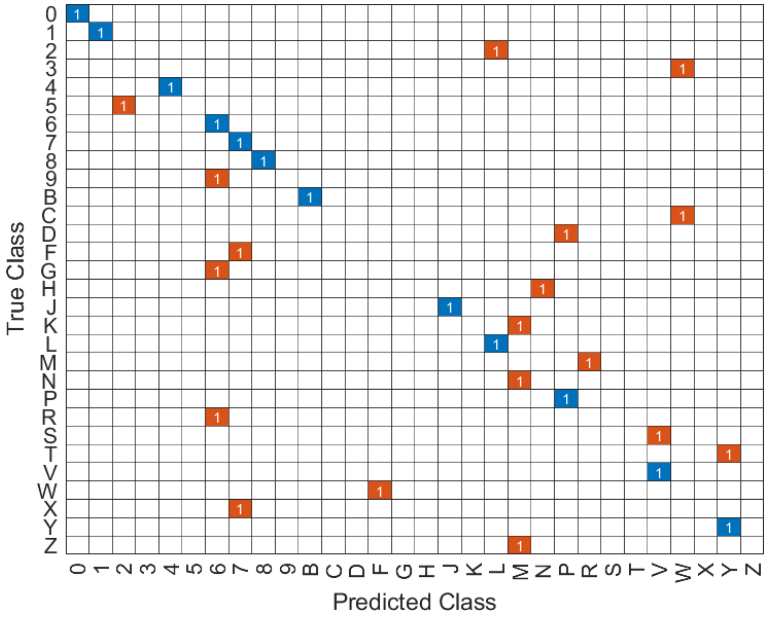
En aquest cas és evident que el resultat és un absolut desastre. Només hem detectat correctament 4 caràcters de 30 (un 13.3 %). És curiós com l'algorisme programat té tendència a considerar els caràcters com a M i W.

Per millorar els resultats anteriors millorarem la selecció de descriptors utilitzat. Eliminem els descriptors C1, C7, C8 i C10. Els descriptors usats en aquesta nova versó son els següents:

```
C2 = prop.Perimeter ./ prop.BoundingBox(:,3);
C3 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C4 = prop.FilledArea ./ prop.Area;
C5 = prop.EulerNumber;
C6 = prop.Eccentricity;
C9 = prop.MaxFeretAngle;
```

Comprovem ara els resultats obtinguts.

Output:



Amb aquesta supressió de descriptors aconseguim millorar notablement l'algorisme. Ara, ha encertat 12 dels 30 caràcters (40%). Tot i que el resultat no és massa bo, s'ha de considerar que les deformacions de la imatge utilitzada son molt importants i que obtenir un resultat gairebé perfecte, utilitzant la imatge sense deformacions, és molt complicat.

Experimentació amb imatges reals

Arribats fins aquí, ens ha semblat interessant observar que passaria si finalment fem prediccions amb imatges que contenen matrícules espanyoles reals.

És clar que les prediccions segurament seran bastant pobres, ja que el nostre classificador és molt senzill, no li hem dotat de molt coneixement, i a més, les característiques no són les òptimes. Però vegem igualment el que passa.

Veurem el que passa amb dos matrícules diferents. Hem intentat dues imatges on es mostren les matrícules de forma frontal i el més clar possible. Aquestes dues imatges són, respectivament:

Abans de analitzar les prediccions, és important destacar que per obtenir de forma correcta les components connexes dels diversos caràcters de les matrícules, per a cada imatge hem hagut de fer un tophat abans de la binaritzacio (perquè el blanc sigui homogèni) i un imopen després de la binarització (per eliminar desperfectes a la imatge, com les petites rodones les quals són cargols que enganxen la matrícula a la carroceria del cotxe.

Més exactament, per a la primera imatge hem fet:

```
i = rgb2gray(imread('matricula_1.jpg'));
figure
imshow(i);

%Per colors homohenis
a = strel('disk', 128);
i = imtophat(i, a);
figure
imshow(i);

IB = i < 100;
figure
imshow(IB);

%IMOPEN
a = strel('disk', 5);
IB = imopen(IB, a);
figure
imshow(IB);


CC = bwconncomp(IB);
```

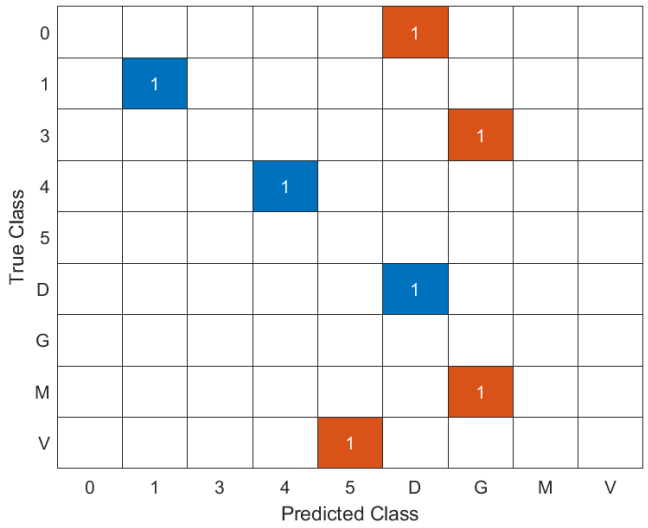
I, per a la segona:

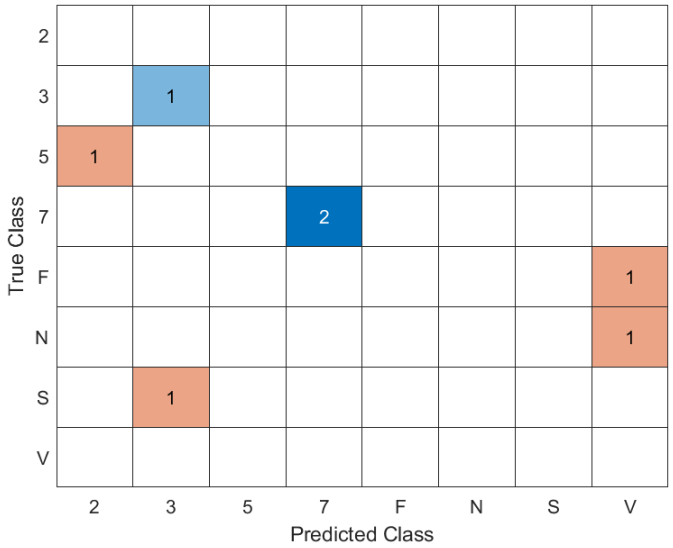
```
i = rgb2gray(imread('matricula_2.jpg'));
figure
imshow(i);

%Per colors homohenis
a = strel('disk', 1000);
i = imtophat(i, a);
figure
imshow(i);

IB = i < 100;
figure
imshow(IB);

%IMOPEN
a = strel('disk', 10);
IB = imopen(IB, a);
figure
imshow(IB);
```

Fet això, ja podem fer les prediccions. Primer vegem que passa si al classificador li passem totes les característiques que hem triat des d'un principi (és a dir, les que hem definit a la introducció). Codi:

```
CC = bwconncomp(IB);
prop = regionprops('table', CC, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'FilledArea', 'Max
C1 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C2 = prop.Perimeter ./ prop.BoundingBox(:,3);
C3 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C4 = prop.FilledArea ./ prop.Area;
C5 = prop.EulerNumber;
C6 = prop.Eccentricity;
C7 = prop.Solidity;
C8 = prop.Extent;
C9 = prop.MaxFeretAngle;
C10 = prop.MinFeretAngle;
X = [C1 C2 C3 C4 C5 C6 C7 C8 C9 C10];
[label, score] = predict(Mdl, X);
```

Observem les "confusionchart" de la primera i de la segona matrícula: respectivament:





Com hem previst, no es tracta de prediccions

gens bones, tot i que encara podrien ser més dolentes. Per a la primera matrícula hem identificar 3/7 caràcters, i per a la segona matrícula també 3/7.

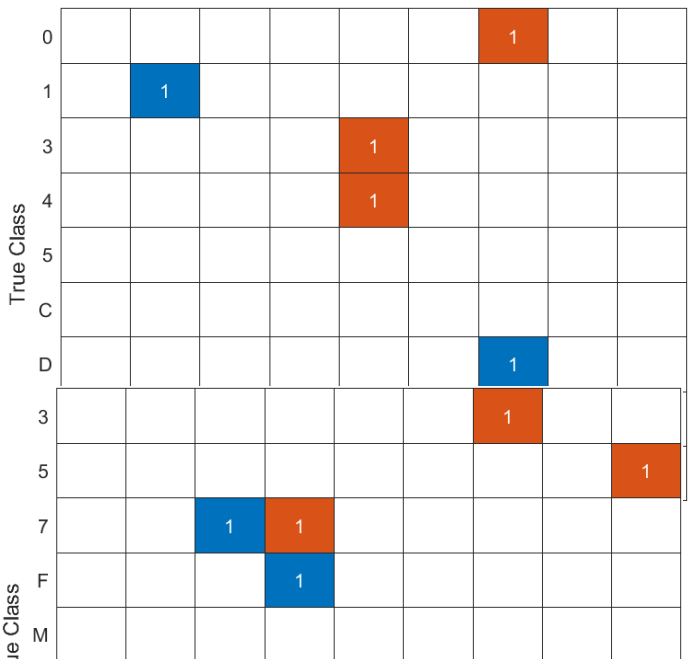
Observem que els caràcters que s'acostumen a detectar correctament són els números, mentre que per a les lletres no se'n detecta casi cap.

Fet això, observem ara que passa si tornem a fer aquestes dues prediccions però ara al classificador només li dotem de les característiques amb les quals hem obtingut més bons resultats per a la segona imatge deformada. Això és interessant ja que en el cas que sigui així llavors podriem confirmar que dotar del classificador només amb aquestes característiques ens proporcionar generalment una presició major.

Recordem, per tant que aquestes característiques són, codi:

```
CC = bwconncomp(IB);
prop = regionprops('table', CC, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'Fille
C1 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C2 = prop.Perimeter ./ prop.BoundingBox(:,3);
C3 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C4 = prop.FilledArea ./ prop.Area;
C5 = prop.EulerNumber;
C6 = prop.Eccentricity;
C7 = prop.Solidity;
C8 = prop.Extent;
C9 = prop.MaxFeretAngle;
C10 = prop.MinFeretAngle;
X = [ C2 C3 C4 C5 C6   C9 ];
[label, score] = predict(Mdl, X);
```

Observem les "confusionchart" per a les prediccions de la primera i segona imatge, respectivament:



Aquesta vegada les prediccions han sigut pitjors. Tant per a la primera com per a la segona matrícula només hem sigut capaços de reconèixer 2/7 caràcters. Per tant, no podem confirmar que el fet de només triar aquestes característiques és generalment millor que respecte com ho hem fet abans.

Finalment, trobem útil observar com serien les prediccions si entrenessim el nostre classificador no només amb les característiques dels caràcters per a una imatge, sinó també amb d'altres mostres.

Aquí, seria molt útil que entrenessim el classificador amb un munt d'imatges de matrícules reals, ja que llavors segurament podem obtenir prediccions notablement bones. Nosaltres, però, ara només ho farem amb les 4 primeres imatges que hem utilitzat en aquesta pràctica, les quals són:

- La original, amb la que portem entrenant el classificador fins ara.
- La mateixa que la original, però sobreescalada.
- La primera imatge deformada.
- La segona imatge deformada.

Aquí el procés de recollir les característiques per als caràcters de les 4 imatges és una mica més tediós, ja que primer hem de recollir les característiques dels caràcters individualment per a cada imatge, i després reagrupar les

característiques d'aquestes 4 imatges per tal de passar-li de forma correcta a "TreeBagger".

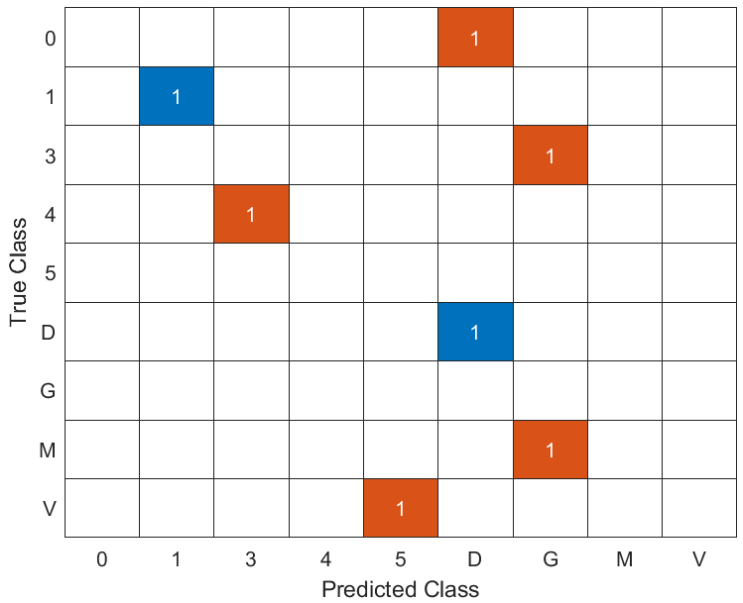
Codi:

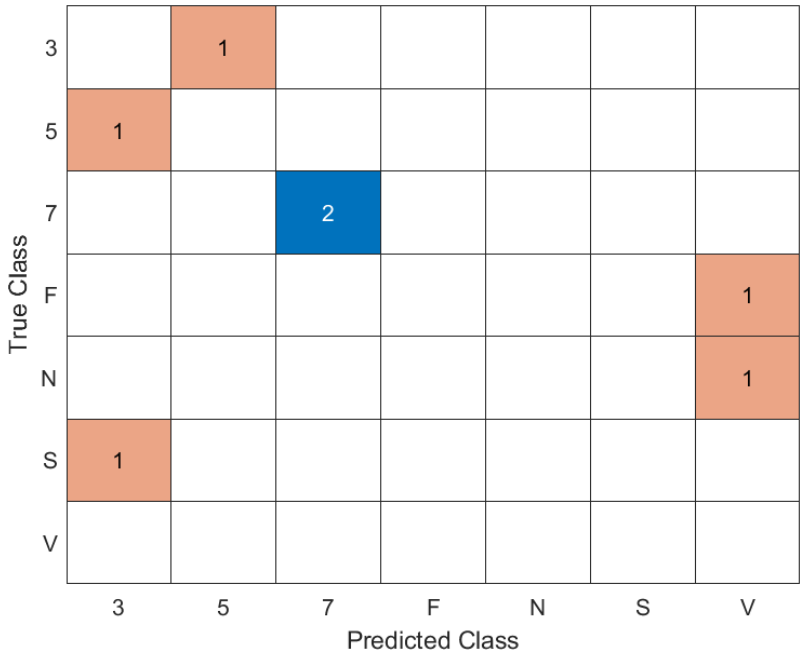
```
i = rgb2gray(imread('joc_de_caracters.jpg'));
figure
imshow(i);
IB = i < 128;
figure
imshow(IB);

ii = rgb2gray(imread('joc_de_caracters.jpg'));
ii = imresize(ii, 2);
IIB = ii < 128;
figure
imshow(IIB);

iii = rgb2gray(imread('Joc_de_caracters_deformats.jpg'));
figure
imshow(iii);
IIIB = iii < 128;
figure
imshow(IIIB);
```

```
iiii = rgb2gray(imread('Joc_de_caracters_deformats_II.png'));
figure
imshow(iiii);
IIIIB = iiii < 128;
figure
imshow(IIIIB);
a = strel('disk', 5);
IIIIB = imopen(IIIIB, a);
figure
imshow(IIIIB);

CC = bwconncomp(IB);
prop = regionprops('table', CC, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'FilledAr
C1 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C2 = prop.Perimeter ./ prop.BoundingBox(:,3);
C3 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C4 = prop.FilledArea ./ prop.Area;
C5 = prop.EulerNumber;
C6 = prop.Eccentricity;
C7 = prop.Solidity;
C8 = prop.Extent;
C9 = prop.MaxFeretAngle;
C10 = prop.MinFeretAngle;



CC2 = bwconncomp(IIB);
prop2 = regionprops('table', CC2, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'FilledA
C21 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C22 = prop.Perimeter ./ prop.BoundingBox(:,3);
C23 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C24 = prop.FilledArea ./ prop.Area;
C25 = prop.EulerNumber;
C26 = prop.Eccentricity;
C27 = prop.Solidity;
C28 = prop.Extent;
C29 = prop.MaxFeretAngle;
C210 = prop.MinFeretAngle;

CC3 = bwconncomp(IIIB);
prop3 = regionprops('table', CC3, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'FilledA
C31 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C32 = prop.Perimeter ./ prop.BoundingBox(:,3);
C33 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C34 = prop.FilledArea ./ prop.Area;
C35 = prop.EulerNumber;
C36 = prop.Eccentricity;
C37 = prop.Solidity;
C38 = prop.Extent;
C39 = prop.MaxFeretAngle;
C310 = prop.MinFeretAngle;
```

```
CC4 = bwconncomp(IIIIB);
prop4 = regionprops('table', CC4, 'Area', 'BoundingBox', 'Perimeter', 'EulerNumber', 'Extent', 'Eccentricity', 'FilledArea'
C41 = prop.Area ./ (prop.Perimeter .* prop.Perimeter);
C42 = prop.Perimeter ./ prop.BoundingBox(:,3);
C43 = prop.BoundingBox(:, 3) ./ prop.BoundingBox(:,4);
C44 = prop.FilledArea ./ prop.Area;
C45 = prop.EulerNumber;
C46 = prop.Eccentricity;
C47 = prop.Solidity;
C48 = prop.Extent;
C49 = prop.MaxFeretAngle;
C410 = prop.MinFeretAngle;

C1 = [C1; C21; C31; C41];
C2 = [C2; C22; C32; C42];
C3 = [C3; C23; C33; C43];
C4 = [C4; C24; C34; C44];
C5 = [C5; C25; C35; C45];
C6 = [C6; C26; C36; C46];
C7 = [C7; C27; C37; C47];
C8 = [C8; C28; C38; C48];
C9 = [C9; C29; C39; C49];
C10 = [C10; C210; C310; C410];

X = [C1 C2 C3 C4 C5 C6 C7 C8 C9 C10];
Y = ['0'; '1'; '2'; '3'; '4'; '5'; '6'; '7'; '8'; '9'; 'B'; 'C'; 'D'; 'F'; 'G'; 'H'; 'J'; 'K'; 'L'; 'M'; 'N'; 'P'; 'R'; 'S'
Mdl = TreeBagger(100, X, Y); %classificador
```

Observem les prediccions per a les dues matrícules si entrenem el classificador d'aquesta manera:





Amb les "confusionchart" vegem com sorprenentment les prediccions tornen a ser pitjors, tot i haver dotat de més informació al nostre classificador. Tot i que al cap i a la fi aquesta informació extra de la qual li hem dotat al classificador tampoc li haurà servit de molt aquest ja que les 4 imatges amb el que l'hem entrenat són molt iguals, els pobres resultats d'aquestes dos prediccions ens porten a pensar que segurament les característiques que hem escollit no són les millors, ni molt menys.