Pràctica 2.
Reconeixement automàtic de dígits manuscrits

Adem Ait

David Agut

Descriptors

Inicialment hem analitzat unes poques imatges i hem extret uns 200 descriptors per cada imatge. Aquests són els següents:

- Descriptors de suma
    - 20 descriptors que sumen el valor de cada píxel per fila
    - 20 descriptors que sumen el valor de cada píxel per columna
- Descriptors de distància
    - 20 descriptors que indiquen la distància entre la vora esquerra i el primer píxel de cada fila
    - 20 descriptors que indiquen la distància entre la vora dreta i l'últim píxel de cada fila
    - 20 descriptors que indiquen la distància entre la vora superior i el primer píxel de cada columna
    - 20 descriptors que indiquen la distància entre la vora inferior i l'últim píxel de cada columna
    - I 80 descriptors més que calculen per cada descriptor de distància la diferència amb la fila o columna anterior

Més endavant hem incorporat els HOG(Histograms of Oriented Gradients). A continuació explicarem més en detall perquè hem triat aquests descriptors.

Descriptors de suma

Aquests descriptors funcionen tal que suma tots els píxels d'una mateixa fila o columna. Per tant es generen 40 descriptors, 20 de fila i 20 de columna.
Hem escollit aquests descriptors perquè són molt senzills d'extreure i ens poden donar bastanta informació sobre el dígit. Aquests descriptors han donat bons resultats. Tot i això només amb aquests descriptors no ha sigut suficient ja que hi ha molta confusió en digits com el 6 i el 8, el 2 i el 5, etc. Però diferencia ben clarament digits com el 1 i el 3 o el 1 i el 8.

Descriptors de distància

Per tal de millorar el percentatge d'encert hem fet recerca i hem trobat uns descriptors que ens han semblat interessants. Aquests són els descriptors de distància. La idea d'aquests descriptors és calcular la distància per cada fila o columna a cada una de les quatre vores: dreta, esquerra, superior i inferior. La distància dels eixos horitzontals es calculen per cada fila, i per calcular la distància a la vora esquerra s'agafa només el píxel més a l'esquerra, i el mateix per la dreta. L'eix vertical funciona amb la mateixa lògica.
Hem pensat que aquests descriptors poden donar força informació sobre la imatge, ja que en certa manera és com calcular el contorn. A més a més són molt simples de calcular.
Per tant teniem 80 descriptors més els 40 que teniem de descriptors de suma. Amb aquests 120 descriptors hem vist un lleuger augment en el percentatge d'encert.

Un cop aquí hem pensat que també, enlloc d'agafar xifres absolutes, podríem treballar amb la diferència entre files o columnes, així veuriem la variació entre elles. D'aquesta manera fa que les xifres siguin menys sensibles a canvis de distància, ja que un mateix número podria estar desplaçat en qualsevol dels eixos. Per tant tindríem uns nous descriptors amb la mateixa informació, no deixen de descriure el contorn, però serien invariants a la traslació.

Descriptors HOG

Hem decidit utilitzar aquests descriptors ja que després de fer recerca hem vist que és un descriptor molt potent i molt popular a l'hora de fer *object recognition*. Pensem que, tot i que s'utilitza més per a reconèixer persones, també pot anar molt bé per reconèixer dígits, ja que es invariant a la rotació i a la il·luminació (tot i que aquesta ultima propietat no s'apliqui gaire per aquest cas, ja que les imatges son en escala de grisos i la il·luminació és pràcticament la mateixa en la gran majoria d'imatges).
A l'hora de decidir quin tamany de la cel·la del HOG utilitzar, hem decidit fer servir una mida de 2x2, ja que les imatges de mostra són bastant petites, de només 20x20. Amb un tamany de cel·la de 2x2 podem capturar els petits detalls de la imatge, que ens permeten identificar quin dígit és. En canvi, si féssim servir una mida més gran, sí que és cert que guanyaríem informació espaial a gran escala, però també perdríem el detall a petita escala. Hem fet servir un tamany de bloc de 2x2 per la mateixa raó d'abans, les imatges són petites. Pel número de bins de l'histograma n'hem fet servir 9, ja que és una mica un punt intermig entre

quantitat adequada d'informació sense perdre massa velocitat de processament, ja que si agaféssim números més grans, tindríem més informació però també ocuparia molt més espai.

A la part d'experimentació s'explica més en detall perquè hem utilitzat cel·les de dimensió 2x2.

Classificadors utilitzats

Support Vector Machine

El classificador que hem utilitzat finalment és el cúbic SVM (Support Vector Machine). La raó per la qual hem decidit utilitzar aquest classificador és perquè ens permet tenir un kernel no lineal, això vol dir que l'algoritme ens permet traçar una divisió entre categories que no és lineal, que ens permet classificar de manera més complexe. No obstant també s'ha de anar amb compte amb l'overfitting. Aquest és un bon classificador, ja que va ser el que van fer servir Navneet Dalal i Bill Triggs en el seu treball sobre HOG per a detectar persones.

La idea del SVM és simple, l'algorisme crea una recta o un hiperplà per a separar les dades en classes. Per tant SVM és un algorisme el qual rep dades i retorna una línia que separa les dades en classes, en el cas que sigui possible. Per obtenir la línia òptima l'algorisme busca el punt, o punts, de cada classe més propers a aquesta. Aquests punts són anomenats *suport vector*. Un cop trobats es calcula la distància entre els punts i la línia. Aquesta distància s'anomena marge. L'objectiu de l'algorisme és maximitzar el marge. Aleshores l'hiperplà el qual té el marge més gran és l'hiperplà òptim.
En el cas que les dades no siguin linealment separables no es pot dibuixar una línia que separa les dades en classes. En aquest cas, les dades poden ser convertides en linealmente separables en una dimensió superior. Un cop aquí es pot dibuixar la línia i mitjançant transformacions matemàtiques, es pot projectar aquesta *decision boundary* a la dimensió original.
Un hiperpla en un espai euclidià n-dimensional, és un subconjunt n-1 dimensional pla d'aquest espai el qual divideix l'espai en dues parts inconnexes.

Linear Discriminant

L'altre classificador que inicialment havíem considerat és el Linear Discriminant. Una de les raons perquè vam considerar aquest classificador, a part dels bons resultats que donava, és que el SVM podria donar problemes d'overfitting.
La base del funcionament del Linear Discriminant és reduir les dimension del nostre set de dades. Això ho fa calculant un nou eix el qual projectar les dades en n+1 dimensions en l'eix de dimensió n.

Aquest nou eix es genera utilitzant dos criteris: maximitzar la distància entre les mitjanes de les classes i minimitzar la variació, la qual el Linear Discriminant anomena *scatter,* que hi ha dins de cada classe.

Experiments realitzats

Per realitzar els experiments i proves hem fet servir la eina *Classification Learner* de MatLab i hem entrenat amb tots els classificadors. Més endavant hem observat que els classificadors *Naive Bayes* trigava molt i donava mals resultats. Per tant aquest tipus de classificador l'hem rebutjat des de l'inici.

Els primers descriptors que vam extreure van ser els descriptors de suma. Amb aquests només, ens donava un 85% d'encert en l'entrenament amb una mostra de 1000 imatges. Com que inicialment ja donaven bons resultats els vam acceptar i vam cercar nous descriptors.

Tot seguit vam extreure els descriptors de distància. Amb aquests vam veure un lleuger augment en el percentatge d'encert. Els primers descriptors de distància, al treballar amb xifres absolutes, eren molt sensibles a la translació. Per tant vam buscar una manera de fer-los més robustos a la translació. D'aquí es van originar els nous descriptors de distància que treballaven amb la diferència de valors per fila o columna. Amb aquests nous descriptors vam fer varies proves permutant els dos tipus de descriptors de distància juntament amb els de suma. La millor combinació era utilitzar els dos tipus de descriptors de distància, el que treballa amb xifres absolutes i el que ho fa amb diferències.

Amb aquests tres tipus de descriptors ens donava un percentatge d'encert del 92% en l'entrenament amb mostra de 1000 imatges.

Per augmentar aquest percentatge vam buscar uns descriptors que ens donessin informació detallada del contorn dels números, ja que és una de les característiques més importants a l'hora de diferenciar xifres. Per extreure aquesta característica vam provar amb els descriptors de HOG, ja descrits en el seu apartat.

Amb aquests nous descriptors, 2916 per imatge, vam veure un augment considerable del percentatge d'encert.

Primer vam intentar provar-ho amb 1000 imatges i ens va donar un 95,4% d'encert. Per veure si aquest percentatge millorava amb més mostres, vam intentar entrenar-lo amb 30000 imatges amb un offset de 15000. Malauradament, no vam poder finalitzar l'execució ja que cap ordinador, tant nostre com el remot de la FIB, no tenia la suficient potència per realitzar-ho. Aleshores vam reduir la mostra a 10000 imatges. Aquesta prova és el resultat final que presentem, ja que ens va donar un 98,2% d'encert en l'entrenament, el percentatge més alt assolit.

Per veure si podiem augmentar més aquest percentatge vam fer servir eines com el PCA o funcions com *fscchi2*, per comprovar que no hi haguessin descriptors irrelevants que afectessin al rendiment i a la taxa d'encert del classificador.

Per fer aquestes proves vam fer servir una mostra de 1000 imatges i amb tots els descriptors, ja que només ens interessava veure si hi havia una millora respecte no fer servir cap eina de reducció de descriptors.

Sense fer servir cap eina ens va donar un 95,2% d'encert. Amb PCA un 94,8%, fent servir 280 descriptors. I utilitzant la funció fscchi2 un 93,2% utilitzant el 25% dels descriptors totals, el percentatge més òptim respecte la relació entre resultats i descriptors.

Per tant vam arribar a la conclusió de que no hi havien descriptors irrelevants, o potser no els suficients per afectar al percentatge d'encert a l'entrenament.

Resultats obtinguts

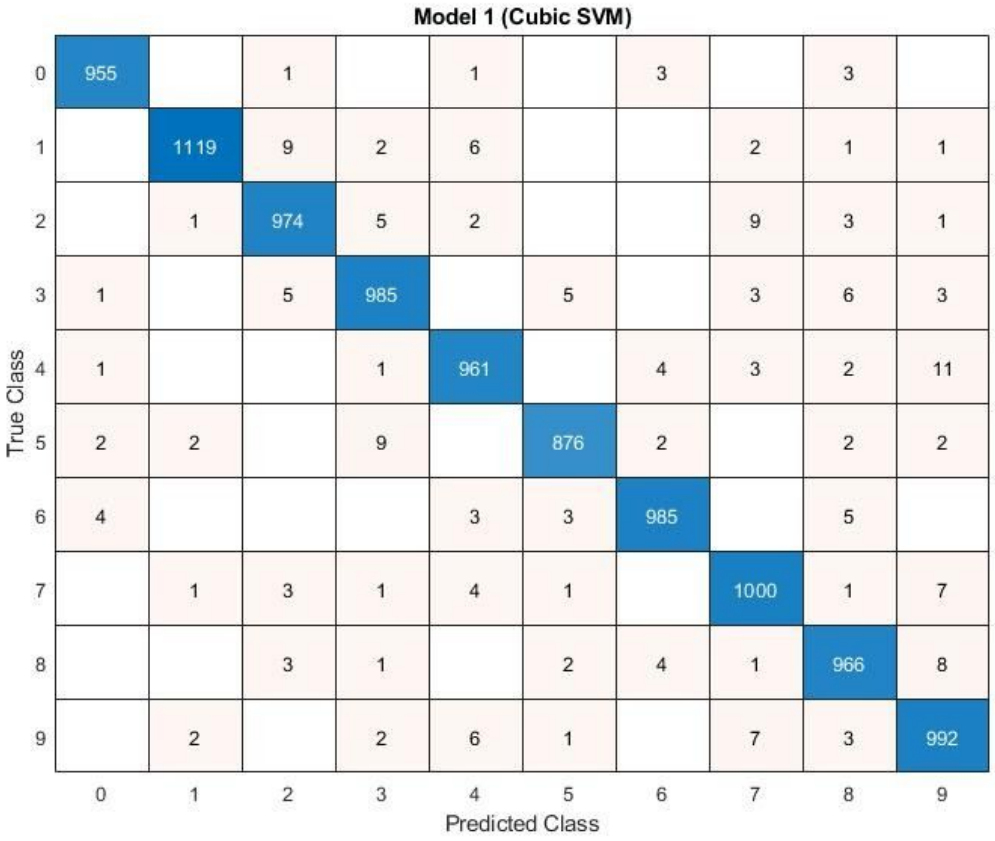
Per obtenir els resultats finals hem utilitzat el classificador Cubic SVM i hem fet servir una mostra de 10.000 imatges. Hem seleccionat un offset de 30.000 seleccionar el subconjunt d'entre totes les imatges. Hem triat aquesta mida per assegurar-nos que el nostre classificador s'entrenava amb una bona varietat d'imatges.

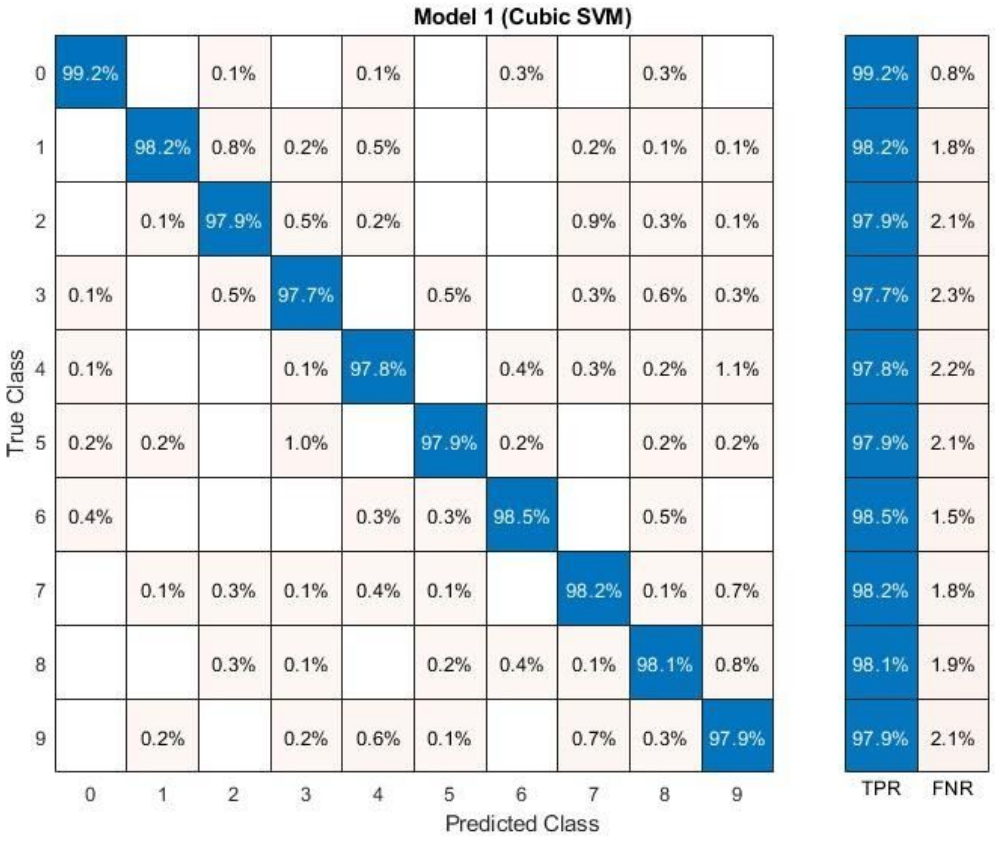
Hem seleccionat els 3112 descriptors obtinguts, ja que després de realitzar vàries proves amb PCA per intentar reduir el número de descriptors, el rendiment del classificador baixava un 1-2%, sense pràcticament baixar temps d'execució, així que hem decidit mantenir-los tots.

Resultats de l'entrenament

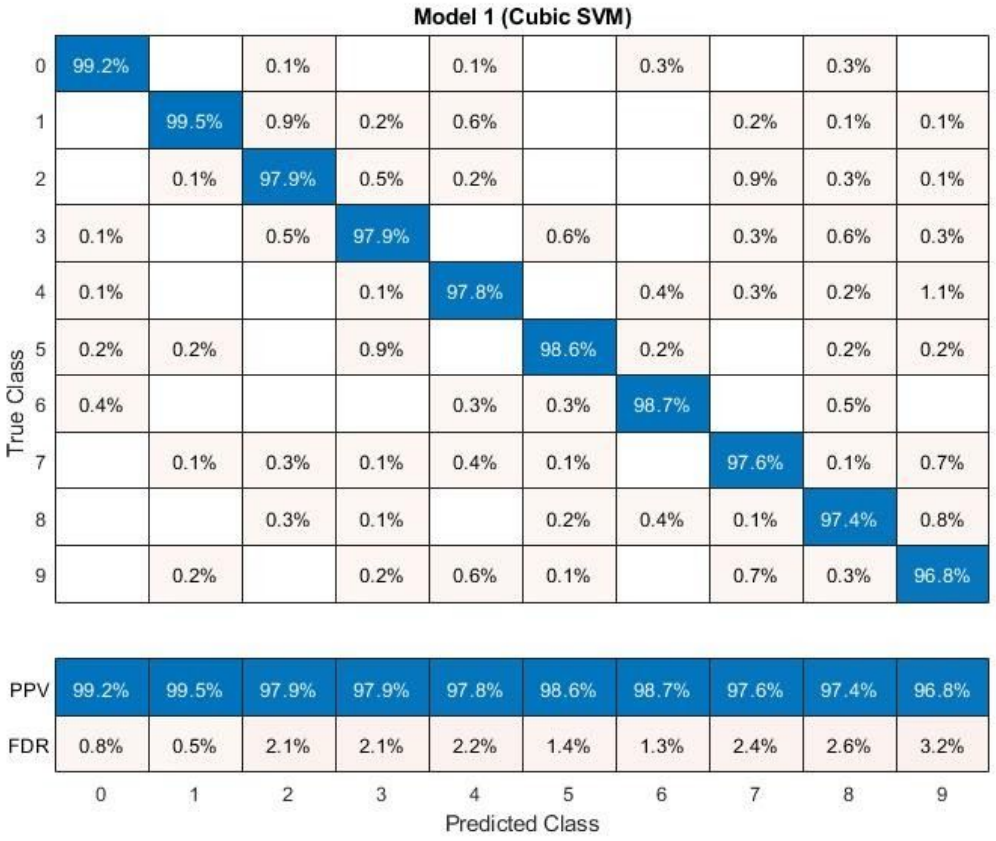
Els resultats obtinguts després de l'entrenament final són els següents.
El rendiment de l'execució final del classificador Cubic SVM ha sigut del 98,2%.

A continuació podem veure la matriu de confusió, amb els True Positive Rate / False Negative Rates i els Positive Predictive Values / False Discovery Rates.


Model 1 (Cubic SVM)
True Class
Predicted Class
0 955 1 1 3 3
1 1119 9 2 6 2 1 1
2 1 974 5 2 9 3 1
3 1 5 985 5 3 6 3
4 1 1 961 4 3 2 11
5 2 2 9 876 2 2 2
6 4 3 3 985 5
7 1 3 1 4 1 1000 1 7
8 3 1 2 4 1 966 8
9 2 2 6 1 7 3 992
0 1 2 3 4 5 6 7 8 9



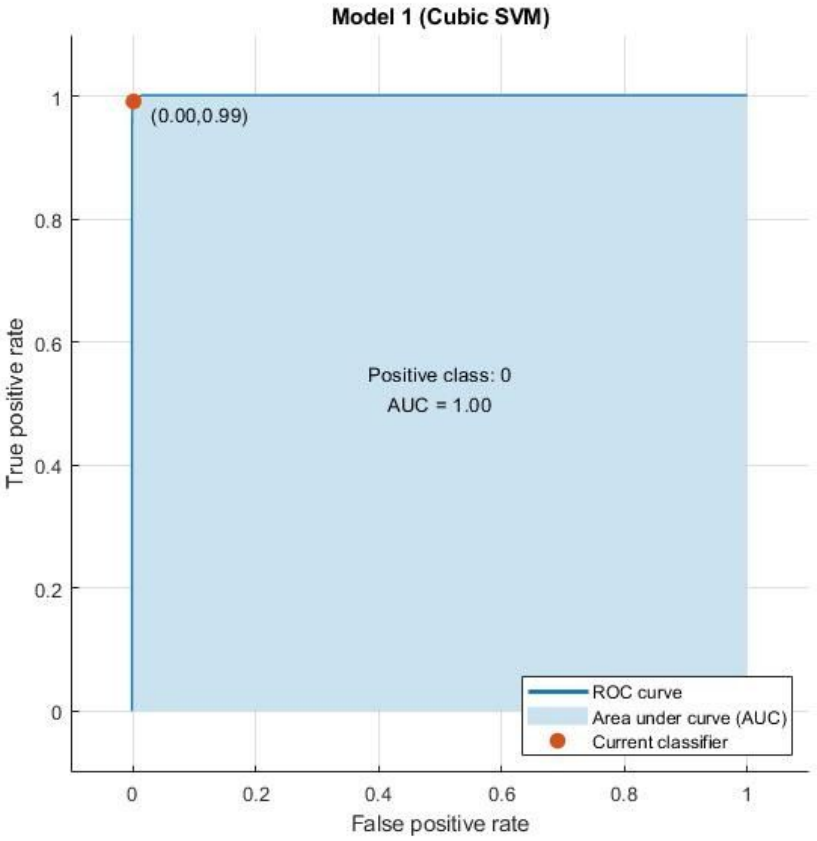
En aquesta taula podem veure el True Positive Rate i el False Negative Rates. Aquests ens indiquen que, de totes les instàncies d'una classe, quines han estat correctament identificades i quines no.



En aquesta taula podem veure el Positive Predictive Values i el False Discovery Rates. Aquests ens indiquen que, de totes les instàncies que hem classificat com una classe, quantes realment eren d'aquella classe.

Podem veure que hem obtingut uns resultat bastant bons. Les classes amb les que el classificador té més problemes són el 9 i el 8.

A continuació tenim la curva de ROC:



Com podem veure, és pràcticament un quadrat, que és el que vindria a ser el resultat ideal.

Resultats de validació

Per obtenir els resultats amb imatges reals, que el classificador no havia vist mai, vam escriure un programa que llegia matrius d'imatges i les seves labels, executava el classificador sobre elles, i després les comparava amb les labels proporcionades per saber si les havia encertat. Els resultats són els següents.

Executant el programa de validació amb un conjunt de 1000 imatges (offset 0) hem obtingut una precisió del 98.1%. Com podem observar, no tenim problemes d'overfitting, que era un dels problemes que podien sorgir utilitzant el kernel no lineal del Cubic SVM.

Name	Value
acc	98.1000
images	*20x20x1000 double*
labels	*1000x1 double*
N	1000
results	*1000x3 table*
timeElapsed	94.8071
trainedModel	*1x1 struct*

En la variable *acc* veiem el resultat de la precisió comentat abans. En la variable *timeElapsed* tenim el temps que ha tardat a realitzar l'execució. En la taula *results* tenim els resultats de les prediccions que ha realitzat el classificador. A continuació tenim un fragment.

	1 PredictedResult	2 RealValue	3 Correcte
1	"5"	"5"	"1"
2	"0"	"0"	"1"
3	"4"	"4"	"1"
4	"1"	"1"	"1"
5	"9"	"9"	"1"
6	"2"	"2"	"1"
7	"1"	"1"	"1"
8	"3"	"3"	"1"
9	"1"	"1"	"1"
10	"4"	"4"	"1"
11	"3"	"3"	"1"
12	"5"	"5"	"1"
13	"3"	"3"	"1"
14	"6"	"6"	"1"
15	"1"	"1"	"1"
16	"7"	"7"	"1"
17	"2"	"2"	"1"
18	"8"	"8"	"1"
19	"6"	"6"	"1"
20	"9"	"9"	"1"

Aquí trobem tres columnes. La primera *PredictedResult* ens indica el resultat que ha predit el classificador. La segona *RealValue* ens indica quin número era realment la imatge (el label). La columna final *Correcte* ens diu si hem encertat la predicció o no.

Funcions utilitzades

Per a realitzar la pràctica hem utilitzat vàries funcions, algunes d'aquestes escrites per nosaltres, i altres de Matlab. Primer tenim les funcions implementades per nosaltres.

Funcions implementades per nosaltres

Funció features.m

Aquesta funció ha estat implementada per nosaltres i s'encarrega d'extreure els descriptors de suma i distància d'una imatge. El paràmetre d'entrada és la imatge de la qual volem extreure els descriptors i els paràmetres de sortida són els diferents descriptors, dividits per tipus i per fila i columna.

Funció getDigit.m

Aquesta funció ha estat implementada per nosaltres, i és la que s'encarrega de provar el rendiment del classificador amb imatges reals. Té dos paràmetres d'entrada obligatoris, que són la matriu d'imatges que vols classificar i el model que es vol fer servir. El model el passem per no haver de carregar-lo abans de cada execució, agilitzant el procés. A més a més, també té un paràmetre opcional. Aquest paràmetre és el de *labels*, que correspon a una matriu dels *labels* de les imatges que s'han passat. S'utilitza per exteure els resultats del classificador i comparar-los amb els reals, i també per veure l'accuracy.
Si no es passa el paràmetre labels, la funció retorna una taula d'una sola columna amb els resultats predits. Si el paràmetre labels està present, la funció retorna un double que conté l'accuracy de les prediccions i una taula. Aquesta taula conté 3 columnes: *PredictedResult*, *RealValue* i *Correcte*, d'squerra a dreta. La primera columna conté els valors predits pel classificador. La segona columna conté els valors reals dels dígits (el contingut dels labels) i la tercera columna conté un booleà que indica si la predicció és correcta o no.

Funció reduirDescriptors.m

Aquesta funció ha estat implementada per nosaltres i s'utilitza conjuntament amb la funció de Matlab *fscchi2* per reduir el nombre de descriptors obtingut. Com a paràmetre d'entrada li hem de passar el array resultant de la funció *fscchi2*, la taula de descriptors, el número de descriptors N que volem conservar i el nombre de descriptors totals. A continuació la funció va eliminant tots els descriptors que no estiguin part dels N millors i ens retorna una nova taula amb els pitjors descriptors eliminats.

Funcions utilitzades de Matlab

A continuació tenim totes les funcions de Matlab o d'altres fonts que han sigut utilitzades durant la pràctica.

Funció zeros

Funció utilitzada per generar arrays on tots els números son zeros.

Funció readMNIST

Funció proporcionada per la pràctica que s'utilitza per llegir els fitxers .idx3-ubyte i idx1-ubyte per extreure les imatges i labels usades per entrenar i validar el classificador.

Funció strings

Funció utilitzada per generar arrays de strings.

Funció table

Funció que utilitzem per generar una taula al nostre gust.

Funció extractHOGFeatures

Funció que extreu els features dels *histogram of oriented gradients.* Hem fet servir aquesta funció per obtenir els descriptors de HOG per entrenar el nostre classificador. Hem utilitzat el paràmetre CellSize 2x2 per definir el tamany de cel·la que ens interessava. Hem triat aquest tamany perquè era un bon intermig entre detalls a nivell local i informació a gran escala i també perquè ens donava els millors resultats. El paràmetre del número de bins l'hem deixat per defecte. Els altres paràmetres també s'han deixat per defecte.

Funció fscchi2

Funció que serveix per ordenar features per un ranking de valor. Hem fet servir aquesta funció per intentar reduïr el nombre de descriptors obtinguts, ja que segurament alguns no eren tan importants com altres. Funciona de la manera següent: es passen com a paràmetres la taula que conté els descriptors i quina és la variable de resposta. La funció retorna un array amb els indexos de la taula ordenats segons la relevància. D'aquesta manera, si la primera posició d'aquest array és el número 34, vol dir que el predictor més important és el de la columna 34 de la taula de predictors.

Funció trainClassifier

Funció que serveix per entrenar el classificador. És l'equivalent a utiltizar el classification learner.

Funció predictFcn

Funció que intenta predir el valor d'un dígit basant-se en els seus classificadors. Aquesta funció l'hem utilitzat per validar el rendiment del classificador amb imatges reals. S'obté a partir de l'extracció del model del classification learner un cop ha estat entrenat.

Bibliografia i referències

Links de recerca consultats.

https://ch.mathworks.com/help/stats/train-decision-trees-in-classification-learner-app.html

https://ch.mathworks.com/help/stats/export-classification-model-for-use-with-new-data.html

https://ch.mathworks.com/help/vision/ref/extracthogfeatures.html#d120e150360

https://es.mathworks.com/help/stats/fscchi2.html

https://www.learnopencv.com/histogram-of-oriented-gradients/

https://medium.com/analytics-vidhya/a-take-on-h-o-g-feature-descriptor-e839ebba1e52

https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

https://www.researchgate.net/publication/331950772_Handwritten_Digit_Recognition_using_Slope_Detail_Features

https://en.wikipedia.org/wiki/Support_vector_machine

https://en.wikipedia.org/wiki/Linear_discriminant_analysis

Annex: Codi

Funció features

Aquesta funció s'encarrega d'extreure els descriptors de suma i distància d'una imatge.

```
function
[vectorSumRow,vectorSumCol,vectorDistRowLeft,vectorDistRowRight,vectorDist
ColUp,vectorDistColDown,vectorDiffRowLeft,vectorDiffRowRight,vectorDiffColUp,
vectorDiffColDown] = features(matrixIm)

[row, col] = size(matrixIm);
vectorSumRow = zeros(1,row);
vectorSumCol = zeros(1,col);

vectorDistRowLeft = zeros(1,row);
vectorDistRowRight = zeros(1,row);
vectorDistColUp = zeros(1,col);
vectorDistColDown = zeros(1,col);

vectorDiffRowLeft = zeros(1,row-1);
vectorDiffRowRight = zeros(1,row-1);
vectorDiffColUp = zeros(1,col-1);
vectorDiffColDown = zeros(1,col-1);

vectorDistRowLeft(1,:) = 400;
vectorDistRowRight(1,:) = 400;
vectorDistColUp(1,:) = 400;
vectorDistColDown(1,:) = 400;

for j = 1:col
   vectorSumCol(j) = sum(matrixIm(:,j));
end
for i = 1:row
   vectorSumRow(i) = sum(matrixIm(i,:));
   for j = 1:col
      if (matrixIm(i,j) > 0)
         if(j < vectorDistRowLeft(1,i))
            vectorDistRowLeft(1,i) = j - 1;
         end
         if((col - j) < vectorDistRowRight(1,i))
            vectorDistRowRight(1,i) = (col - j);
         end
         if(i < vectorDistColUp(1,j))
```

```
            vectorDistColUp(1,j) = i - 1;
        end
        if((row - i) < vectorDistColDown(1,j))
            vectorDistColDown(1,j) = (row - i);
        end
    end
  end
end

for i = 1:row-1
  vectorDiffRowLeft(1,i) = vectorDistRowLeft(1,i+1) - vectorDistRowLeft(1,i);
  vectorDiffRowRight(1,i) = vectorDistRowRight(1,i+1) -
vectorDistRowRight(1,i);
  vectorDiffColUp(1,i) = vectorDistColUp(1,i+1) - vectorDistColUp(1,i);
  vectorDiffColDown(1,i) = vectorDistColDown(1,i+1) - vectorDistColDown(1,i);
end

end
```

Funció reduirDescriptors.m

Aquesta funció elimina un determinat nombre de descriptors de la taula de descriptors.

```
function [TResult] = reduirDescriptors(idx,T,numFeat,numTotal)

start = numFeat;
final = numTotal;
TResult = T;
for i = start:final
    var = 'Var';
    r = strcat(var,int2str(idx(i)));
    TResult = removevars(TResult,r);
end
end
```

Programa MATLAB d'entrenament

En aquesta secció trobem el codi corresponent a obtenir tots els descriptors de les imatges per entrenar el classificador.

```
%Lectura matriu de digits
N=100;
[images, labels] =readMNIST( ...
    'train-images-idx3-ubyte\train-images.idx3-ubyte', ...
    'train-labels-idx1-ubyte\train-labels.idx1-ubyte',N,0);
```

```
%Extraccio dels descriptors de totes les imatges
Nft = 3112;
count = 0;
vartypes = strings([1,Nft+1]);
vartypes{Nft+1} = 'string';
vartypes(1:Nft) = 'double';
T = table('Size',[N Nft+1],'VariableTypes',vartypes);


for i = 1:N
    im=images(:,:,i);
    imshow(im)
    [sumRow,sumCol,distRowLeft,distRowRight,distColUp,distColDown, ...
        diffRowLeft,diffRowRight,diffColUp,diffColDown] = features(im);

    T{i,1:20} = sumRow;
    T{i,21:40} = sumCol;
    T{i,41:60} = distRowLeft;
    T{i,61:80} = distRowRight;
    T{i,81:100} = distColUp;
    T{i,101:120} = distColDown;
    T{i,121:139} = diffRowLeft;
    T{i,140:158} = diffRowRight;
    T{i,159:177} = diffColUp;
    T{i,178:196} = diffColDown;

    [hog_2x2, vis2x2] = extractHOGFeatures(im,'CellSize',[2 2]);
    T{i,197:3112} = hog_2x2;
    T.Var3113(i)=labels(i);
end
```

```
%Reduccio de descriptors
Nft = 3112;
idx = fscchi2(T,'Var3113');
Nred = 0.25 * Nft;
Nred = int16(Nred);
T2 = reduirDescriptors(idx,T,Nred,Nft);
```

```
%Entrenar el classificador programaticament
[trainedClassifier, validationAccuracy] = trainClassifier(T)
```

Programa MATLAB de validació

En aquesta secció trobem el codi que valida el classificador amb imatges que no ha vist mai.

```
function [outputArg2, outputArg1] = getDigit(trainedModel, matriuImatges,
labels)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

N = size(matriuImatges,3);

Nft = 3112;
vartypes = strings([1,Nft+1]);
vartypes{Nft+1} = 'string';
vartypes(1:Nft) = 'double';
T = table('Size',[N Nft+1],'VariableTypes',vartypes);

for i = 1:N
   im=matriuImatges(:,:,i);

[sumRow,sumCol,distRowLeft,distRowRight,distColUp,distColDown,diffRowLeft,di
ffRowRight,diffColUp,diffColDown] = features(im);

   T{i,1:20} = sumRow;
   T{i,21:40} = sumCol;
   T{i,41:60} = distRowLeft;
   T{i,61:80} = distRowRight;
   T{i,81:100} = distColUp;
   T{i,101:120} = distColDown;
   T{i,121:139} = diffRowLeft;
   T{i,140:158} = diffRowRight;
   T{i,159:177} = diffColUp;
   T{i,178:196} = diffColDown;

   [hog_2x2, vis2x2] = extractHOGFeatures(im,'CellSize',[2 2]);
   T{i,197:3112} = hog_2x2;

end
```

```
yfit = trainedModel.predictFcn(T);
if ~exist('labels','var')
    T.Var3113(:) = "?";
    error = 0;
    outputArg2 = table('Size',[N
1],'VariableTypes',{'string'},'VariableNames',{'PredictedResult'});
    for i = 1:N
        outputArg2{i,1} = string(yfit(i));
    end
    outputArg1 = 0;
else
    T.Var3113(:)=labels(:);
    error = 0;
    outputArg2 = table('Size',[N 3],'VariableTypes',{'string' 'string'
'string'},'VariableNames',{'PredictedResult' 'RealValue' 'Correcte'});
    for i = 1:N
        outputArg2{i,3} = "1";
        if (string(yfit(i)) ~= T.Var3113(i))
            error = error + 1;
            outputArg2{i,3} = "0";
        end
        outputArg2{i,1} = string(yfit(i));
        outputArg2{i,2} = T.Var3113(i);
    end
    outputArg1 = (1 - (error / N)) * 100;
    end
end
```