**BÁO CÁO THỰC HÀNH**

**MÔN NHẬN DANG THỊ GIÁC VÀ ỨNG DỤNG**

**BÀI TẬP THỰC HÀNH 1**

Họ và tên: Lê Hoàng Dũng

MSHV: CH1501022

Github: <https://github.com/lehoangdung0612/VRA.LeHoangDung.CH1501022>

**BÀI TẬP THỰC HÀNH 2**

Q1. Hãy cho biết câu lệnh tạo ngẫu nhiên một số trong đoạn [1 200]

**>> randi([1 200]);**

Q2. Truy cập phần tử dòng 3 cột 5 của ma trận A

**>> A(3, 5);**

Q3. Lệnh tạo ma trận A kích thước 100x200 và các giá trị đều là zero (0).

**>> A = zero(100,200);**

Q4. Lệnh lấy số dòng của ma trận A

**>> size(A, 1);**

Q5. Lệnh lấy vector cột 10 của ma trận A

**>> A(:, 10);**

Q6. Lệnh lấy vector dòng 10 của ma trận A

**>> A(10, :);**

Q7. Lệnh chuyển vector dòng thành ma trận 28x28

**>> reshape(vect, 28, 28);**

**BÀI TẬP THỰC HÀNH 3**

**Q1.**

function showTrainImageAtN(n)

if ~exist('n','var')

disp('Wrong parameters...');

return

end

fprintf('\n Load du lieu train);

imgAll = loadMNISTImages('./train-images.idx3-ubyte');

lblAll = loadMNISTLabels('./train-labels.idx1-ubyte');

figure,

img = imgAll(:, n);

img2D = reshape(img, 28, 28); % reshape

strLabelImage = ['Label:', num2str(lblAll(n))];

imshow(img2D); % show image

title(strLabelImage);

end

|  |  |  |
| --- | --- | --- |
| N = 1 | N = 500 | N = 5000 |
| N = 10000 | N = 59000 |  |

**Q2.**

function showTestImageAtN(n)

if ~exist('n','var')

disp('Wrong parameters...');

return

end

fprintf('\n Load du lieu test');

imgAll = loadMNISTImages('./t10k-images.idx3-ubyte');

lblAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');

figure,

img = imgAll(:, n);

img2D = reshape(img, 28, 28); % reshape

strLabelImage = ['Label:', num2str(lblAll(n))];

imshow(img2D); % show image

title(strLabelImage);

end

|  |  |  |
| --- | --- | --- |
| N = 1 | N = 500 | N = 5000 |
| N = 9000 |  |  |

**Q3.**

function thongKeImageTrain()

fprintf('\n Load du lieu');

lblAll = loadMNISTLabels('./train-labels.idx1-ubyte');

nCol = 10;

A = zeros([2 nCol]);

for i=1:nCol

label = i - 1;

A(1, i) = label;

A(2, i) = sum(lblAll == label);

end

% print A

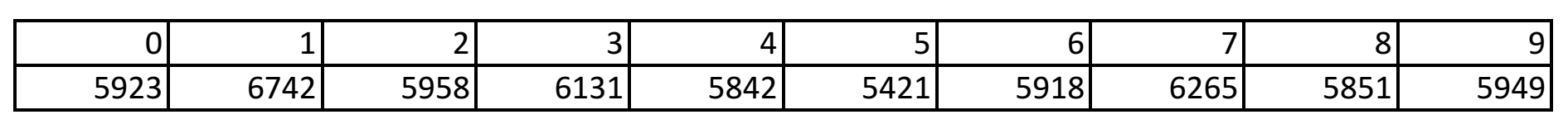
A

% write csv file

strFileName = ['D:\Q3', '.csv'];

csvwrite(strFileName, A);

end



**Q4.**

function thongKeImageTest()

fprintf('\n Load du lieu');

lblAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');

nCol = 10;

A = zeros([2 nCol]);

for i=1:nCol

label = i - 1;

A(1, i) = label;

A(2, i) = sum(lblAll == label);

end

% print A

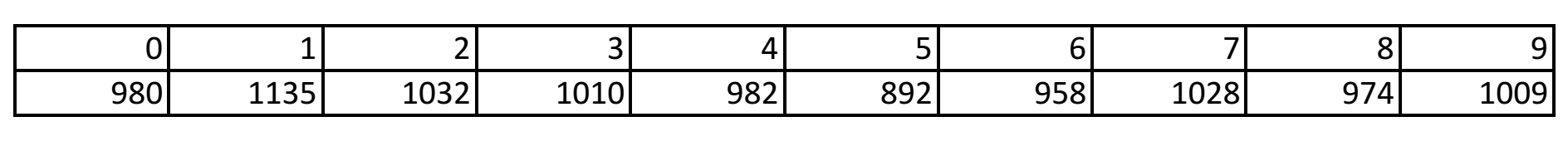
A

% write csv file

strFileName = ['D:\Q4', '.csv'];

csvwrite(strFileName, A);

end



**Q5.**

function lblPredictTest = recognizeImageAtN(n)

if ~exist('n','var')

disp('Wrong parameters...');

return

end

imgTrainAll = loadMNISTImages('./train-images.idx3-ubyte');

lblTrainAll = loadMNISTLabels('./train-labels.idx1-ubyte');

Mdl = fitcknn(imgTrainAll', lblTrainAll);

imgTestAll = loadMNISTImages('./t10k-images.idx3-ubyte');

imgTest = imgTestAll(:, n);

lblPredictTest = predict(Mdl, imgTest');

figure;

img2D = reshape(imgTest, 28, 28);

imshow(img2D);

strLabelImage = ['Ket qua nhan dang: ', num2str(lblPredictTest)];

title(strLabelImage);

end

|  |  |  |
| --- | --- | --- |
| N = 5 | N = 500 | N = 900 |

**Q6.**

function checkRecognitionImage(n)

if ~exist('n','var')

disp('Wrong parameters...');

return

end

imgTrainAll = loadMNISTImages('./train-images.idx3-ubyte');

lblTrainAll = loadMNISTLabels('./train-labels.idx1-ubyte');

Mdl = fitcknn(imgTrainAll', lblTrainAll);

imgTestAll = loadMNISTImages('./t10k-images.idx3-ubyte');

lblTestAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');

imgTest = imgTestAll(:, n);

lblPredictTest = predict(Mdl, imgTest');

lblImageTest = lblTestAll(n);

figure;

img2D = reshape(imgTest, 28, 28);

imshow(img2D);

strLabelImage = 'Ban dau ';

strLabelImage = [strLabelImage, num2str(lblTestAll(n)), '.'];

strLabelImage = [strLabelImage, ' Du doan: '];

strLabelImage = [strLabelImage, num2str(lblPredictTest), '.'];

if (lblPredictTest == lblImageTest)

strLabelImage = [strLabelImage, ' Ket qua dung.'];

else

strLabelImage = [strLabelImage, ' Ket qua sai.'];

end

title(strLabelImage);

end

**Q7.**

function countNumOfLabelWrongKnn(n)

if ~exist('n','var')

disp('Wrong parameters...');

return

end

imgTrainAll = loadMNISTImages('./train-images.idx3-ubyte');

lblTrainAll = loadMNISTLabels('./train-labels.idx1-ubyte');

Mdl = fitcknn(imgTrainAll', lblTrainAll);

imgTestAll = loadMNISTImages('./t10k-images.idx3-ubyte');

lblTestAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');

nNumbers = size(imgTestAll, 2);

counter = 0;

for i=1:nNumbers

lblTest = lblTestAll(i);

if num2str(lblTest) == num2str(n)

imgTest = imgTestAll(:, i);

lblPredictTest = predict(Mdl, imgTest');

if lblPredictTest ~= lblTest

counter = counter + 1;

end

end

end

fprintf('So luong anh co label nhan dang sai: %d\n', counter);

end

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|  | 7 | 6 | 40 | 40 | 38 | 32 | 14 | 36 | 54 | 42 |

**Q7\*.**

function createConfusionMatrixKnn()

imgTrainAll = loadMNISTImages('./train-images.idx3-ubyte');

lblTrainAll = loadMNISTLabels('./train-labels.idx1-ubyte');

Mdl = fitcknn(imgTrainAll', lblTrainAll);

imgTestAll = loadMNISTImages('./t10k-images.idx3-ubyte');

lblTestAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');

nNumbers = size(imgTestAll, 2);

nCol = 10;

confusionMatrix = zeros(nCol, nCol);

for i=1:nNumbers

lblTest = lblTestAll(i);

imgTest = imgTestAll(:, i);

lblPredictTest = predict(Mdl, imgTest');

confusionMatrix(lblTest + 1, lblPredictTest + 1) = confusionMatrix(lblTest + 1, lblPredictTest + 1) + 1;

end

disp('Confusion matrix');

confusionMatrix

% write csv file

strFileName = ['D:\Q71', '.csv'];

csvwrite(strFileName, confusionMatrix);

end

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Predicted Label | | | | | | | | | |
| Actual Label Test |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | **973** | 1 | 1 | 0 | 0 | 1 | 3 | 1 | 0 | 0 |
| 1 | 0 | **1129** | 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 7 | 6 | **992** | 5 | 1 | 0 | 2 | 16 | 3 | 0 |
| 3 | 0 | 1 | 2 | **970** | 1 | 19 | 0 | 7 | 7 | 3 |
| 4 | 0 | 7 | 0 | 0 | **944** | 0 | 3 | 5 | 1 | 22 |
| 5 | 1 | 1 | 0 | 12 | 2 | **860** | 5 | 1 | 6 | 4 |
| 6 | 4 | 2 | 0 | 0 | 3 | 5 | **944** | 0 | 0 | 0 |
| 7 | 0 | 14 | 6 | 2 | 4 | 0 | 0 | **992** | 0 | 10 |
| 8 | 6 | 1 | 3 | 14 | 5 | 13 | 3 | 4 | **920** | 5 |
| 9 | 2 | 5 | 1 | 6 | 10 | 5 | 1 | 11 | 1 | **967** |

**Q8\*\*.**

function calculatePrecisionOfKnn(NumNeighbors, Distance)

% NumNeighbors Number of neighbors to predict

% Distance 'euclidean'/ 'seuclidean'/ 'cityblock'/ 'chebychev'/ 'minkowski'

% / 'mahalanobis'/ 'cosine'/ 'correlation'/ 'spearman'/ 'hamming'/ 'jaccard'

if ~exist('NumNeighbors','var')

NumNeighbors = 1;

end

if ~exist('Distance','var')

Distance = 'euclidean';

end

imgTrainAll = loadMNISTImages('./train-images.idx3-ubyte');

lblTrainAll = loadMNISTLabels('./train-labels.idx1-ubyte');

Mdl = fitcknn(imgTrainAll', lblTrainAll, 'NumNeighbors', NumNeighbors, 'Distance', Distance);

imgTestAll = loadMNISTImages('./t10k-images.idx3-ubyte');

lblTestAll = loadMNISTLabels('./t10k-labels.idx1-ubyte');

nNumbers = size(imgTestAll, 2);

nCol = 10;

confusionMatrix = zeros(nCol, nCol);

for i=1:nNumbers

lblTest = lblTestAll(i);

imgTest = imgTestAll(:, i);

lblPredictTest = predict(Mdl, imgTest');

confusionMatrix(lblTest + 1, lblPredictTest + 1) = confusionMatrix(lblTest + 1, lblPredictTest + 1) + 1;

end

precision = 0;

recall = 0;

accuracy = 0;

for i=1:nCol

accuracy = accuracy + confusionMatrix(i, i);

precision = precision + confusionMatrix(i, i) / sum(confusionMatrix(i, :));

recall = recall + confusionMatrix(i, i) / sum(confusionMatrix(:, i));

end

accuracy = 100 \* accuracy / nNumbers;

precision = 100 \* precision / nCol;

recall = 100 \* recall / nCol;

fprintf('\nAccuracy= %s', num2str(accuracy));

fprintf('\nPrecision= %s', num2str(precision));

fprintf('\nRecall= %s\n', num2str(recall));

end

>> calculatePrecisionOfKnn(1, 'euclidean')

Accuracy= 96.91

Precision= 96.8794

Recall= 96.9148

>> calculatePrecisionOfKnn(1, 'cosine')

Accuracy= 97.23

Precision= 97.1898

Recall= 97.238

>> calculatePrecisionOfKnn(3, 'euclidean')

Accuracy= 97.06

Precision= 97.0273

Recall= 97.0921

>> calculatePrecisionOfKnn(3, 'cosine')

Accuracy= 97.37

Precision= 97.3384

Recall= 97.377