**KNN**

Program was tested for **3300** test images with **60000** training images. It took 3 hours to give results.

Some Results are given below:

K=3

No of test images=100

No of training images=10000

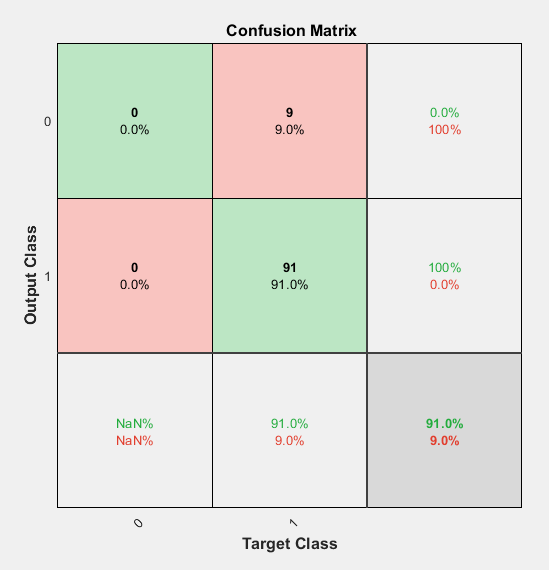
Percentage of Accuracy by Eucidean Distance=95

Percentage of Accuracy by Cosine Similarity=95

Recall by Cosine Similarity= 0.9532

Precision by Cosine Similarity= 0.9559

F\_score by Cosine Similarity=0.9545

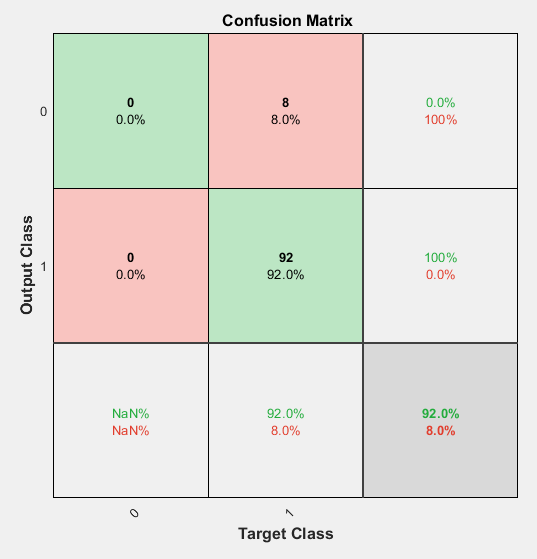
Confusion Matrix of Cosine Similarity

Recall by Eucidean Distance = 0.9497

Precision by Eucidean Distance =0.9644

F\_score by Eucidean Distance = 0.9570

Confusion Matrix of Eucidean Distance



**Code**

Main.m

TestImages=csvread('MNIST/test.csv', 0, 0);

TrainingImages=csvread('MNIST/train.csv', 0, 0);

correctPredictionCountByEucideanDist=0;

correctPredictionCountByCosSimilarity=0;

noOfTestImages=size(TestImages,1);

CorrectLabelArray = zeros(1,noOfTestImages);

CalculatedLabelArrayEucidean = zeros(1,noOfTestImages);

CalculatedLabelArrayCosine = zeros(1,noOfTestImages);

for i=1:noOfTestImages

singleTestImage = zeros(1,784);

singleTestImage(1,:) = TestImages(i,2:785);

%copying every dimension

k=3;

NeighboursID=ClassifyByEuclideanDistance(singleTestImage,TrainingImages,k);

Labels= zeros(1,k);

Labels(1,1:k)=TrainingImages(NeighboursID(1:k),1);

%getting corresponding labels

Majority=mode(Labels);

disp('By Euclidean Distance predicted label');

disp(Majority);

CalculatedLabelArrayEucidean(1,i) = Majority;

if(Majority==TestImages(i,1))

correctPredictionCountByEucideanDist=correctPredictionCountByEucideanDist+1;

end

NeighboursID=ClassifyBySimilarityCosine(singleTestImage,TrainingImages,k);

Labels= zeros(1,k);

Labels(1,1:k)=TrainingImages(NeighboursID(1:k),1);

% getting corresponding labels

Majority=mode(Labels);

disp('By Similarity Cosine predicted label');

disp(Majority);

CalculatedLabelArrayCosine(1,i) = Majority;

disp('actual label');

disp(TestImages(i,1));

CorrectLabelArray(1,i) = TestImages(i,1);

if(Majority==TestImages(i,1))

correctPredictionCountByCosSimilarity=correctPredictionCountByCosSimilarity+1;

end

end

disp('Percentage of Accuracy by Eucidean Distance');

percentage=(correctPredictionCountByEucideanDist/noOfTestImages)\*100;

disp(percentage);

disp('Percentage of Accuracy by Cosine Similarity');

percentage=(correctPredictionCountByCosSimilarity/noOfTestImages)\*100;

disp(percentage);

%plotconfusion(CorrectLabelArray,CalculatedLabelArrayCosine);

plotconfusion(CorrectLabelArray,CalculatedLabelArrayEucidean);

[confMat,order] = confusionmat(CorrectLabelArray,CalculatedLabelArrayEucidean);

%[confMat,order] = confusionmat(CorrectLabelArray,CalculatedLabelArrayCosine);

%%% recall

for i =1:size(confMat,1)

recall(i)=confMat(i,i)/sum(confMat(i,:));

end

recall(isnan(recall))=[];

Recall=1;

Recall=Recall\*(sum(recall)/size(confMat,1));

disp('Recall');

disp(Recall);

%%% précision

for i =1:size(confMat,1)

precision(i)=confMat(i,i)/sum(confMat(:,i));

end

Precision=sum(precision)/size(confMat,1);

disp('Precision');

disp(Precision);

%%% F-score

F\_score=2\*Recall\*Precision/(Precision+Recall); %%F\_score=2\*1/((1/Precision)+(1/Recall));

disp('F\_score');

disp(F\_score);

function neighborIds = ClassifyByEuclideanDistance(TestImage, TrainingMatrix, k)

neighborIds = zeros(1,k);

numOfTrainingImages =size(TrainingMatrix,1);

dist = zeros(2,numOfTrainingImages);

%dist array contains the distances and corresponding indexes

for i=1:numOfTrainingImages

singleTrainingImage = zeros(1,784);

singleTrainingImage(1,:) = TrainingMatrix(i,2:785);

dist(1,i) = EucideanDistance(TestImage,singleTrainingImage);

dist(2,i) = i;

end

[sortedDist correspondingIndexes]=sort(dist(1,:));

neighborIds(1,1:k)=correspondingIndexes(1,1:k);

end

function neighborIds = ClassifyBySimilarityCosine(TestImage, TrainingMatrix, k)

neighborIds = zeros(1,k);

numOfTrainingImages =size(TrainingMatrix,1);

dist = zeros(2,numOfTrainingImages);

%dist array contains the distances and corresponding indexes

for i=1:numOfTrainingImages

singleTrainingImage = zeros(1,784);

singleTrainingImage(1,:) = TrainingMatrix(i,2:785);

dist(1,i) = CosineSimilarity(TestImage,singleTrainingImage);

dist(2,i) = i;

end

[sortedDist correspondingIndexes]=sort(dist(1,:),'descend');

neighborIds(1,1:k)=correspondingIndexes(1,1:k);