IN4085 - Pattern Recognition Final Assignment - Report

Group 17

Mohammad Riftadi
4743792
TU Delft
M.Riftadi@student.tudelft.nl

Enreina Annisa Rizkiasri
4701224
TU Delft
E.A.Rizkiasri@student.tudelft.nl

Nivedita Prasad
4712099
TU Delft
N.Prasad@student.tudelft.nl

ABSTRACT

This is the report for the Final Assignment in Pattern Recognition for GROUP 17.

KEYWORDS

Handwritten numbers, Classification, Clustering

: Group 17

1 INTRODUCTION

This report is written for Automatic Bank Cheque Processing application for our client. We have to automatically classify numbers, which belongs to account numbers and monetary amount. Hence we are using various classification techniques to classify the numbers and compare the techniques to find out the best classification technique to reduce the error rate in classification. We have been given NIST (US National Institute of Standards & Technology) dataset to construct the classifiers and test for two scenarios. There are 10 classes each for one digit from 0 till 9 in the NIST dataset with 2800 handwritten objects for each class. The images scanned are 128 x 128 pixels and in black and white. The first scenario is to train the classifier only once on a large dataset of 200 to 1000 objects per class whereas the second scenario is to use a classifier that is trained for each batch of the cheques that has to be processed on a smaller dataset of 10 objects per class.

We have used input representations of pixel, features, profiles, Zernike moments and dissimilarity measure. For each input representation, we discuss in a detailed manner (1) what is the possible feature reduction and number of features used, (2) the optimal classifier and its type (parametric - nmc, ldc, qdc, fisherc, loglc or non-parametric (knnc, parzenc) or advanced (neural networks, support vector classifiers, one-class classifiers, combining classifiers) (3) the estimated performance of the optimal classifiers on training data.

We finally test our system on a set of benchmark data. We found that our system achieves an error rate of **2.5% for Scenario 1** and an error rate of **23.1% for Scenario 2**.

We have also included a live test to give the client more confidence on the classification system we have built and we have compared it to the benchmark expectations and the performance of our dataset.

The rest of the report is organized as follows: Section 2 consists of detailed description of problem scenarios, Section 3 consists of detailed explanation of dataset and the choice of representation of input features, Section 4 contains explanation of the preprocessing that we have applied, Section 5 briefly describe the feature reduction step, Section 6 briefly describe each classifiers used in our experiment, Section 7 contains experimental results, Section 8 discusses the live test, and finally Section 9 and 10 concludes our experiment and also introduce several recommendations to improve the system.

2 PROBLEM SCENARIOS

As stated in the introduction, our digit recognition classification system is to be built on two different scenarios which varied between the number of available samples that would be used for training our final classifier. This section will briefly discuss the details and challenges of each scenario.

2.1 Scenario 1

The first scenario allows us to use a large training set to build our classifier. Specifically, the scenario encourages us to use as much as data as possible with at least 2000 objects (200 objects per class) and at most 10000 objects (1000 objects per class).

Although the ideal thing to do is to use as much as data for training our classifier, to experiment and estimate our error rate with such big data will be consumptive of both time and memory. As we need to do parameter tuning and cross-validation for estimating the performance of different classifiers, we decided to only use **200 objects per class** for the experiment setup.

After the experiment shows us which classifier (or combination of them) is potentially the best on novel data, we retrain the classifier using the whole dataset available (in this case 100 objects per class) and evaluate this final classifier on the benchmark dataset. As the project's requirement, we are expecting this final classifier to achieve less than or equal to 5% error rate.

2.2 Scenario 1

In the other hand, Scenario 2 only allows us to use much less data compared to what the first scenario allows. This scenario challenges us to use at most 100 objects (10 objects per class) to train our final classifier.

As the maximum size of the training set is already small enough, for the second scenario, we decide to use all samples for both the experiment phase and the phase of building the final classifier. Compared to the first scenario, we expect the estimate error rate will be close to the true error rate of the final classifier on the benchmark test set. The running time of various experiments for this scenario is also faster compared to the first scenario.

The final classifier is to be built similarly to the first scenario, where we first experiment with various classifier and decide which one suits the best to be tested on the benchmark test set. Finally, as required by the client, we are expecting the final classifier for the second scenario to achieve less than or equal to 25% error rate.

3 DATASET AND CHOICE OF REPRESENTATION

In this section we will briefly discuss about the available dataset for training our digit recognition system and what is the available choice of representation. We discuss how each representation is extracted from the original dataset and explain the detail of each feature (if there is any).

The dataset which is supplied from the client is a standard dataset of handwritten digits from the US National Institute of Standards & Technology (NIST)[1]. This dataset in total consists of 2800 images for each 10 digits from "0", "1", and so on until digit "9". Each image is originally scanned from handwritten filled forms and thresholded to become black-and-white images. The images are then segmented into image region of 128x128 pixel, but with adjusted bounding box in order to minimize the size of the dataset.

For this project, we consider three types of representation of this dataset: raw pixel representation, features representation, and dissimilarities representation. We discuss each in detail in the following sections.

3.1 Pixels

The basic and most straightforward representation to be used to train our classifier is the raw pixel representation. Each digit image is represented in a 128x128 matrix (or other adjusted size in that matter) with each element represent the color of the pixel of the said image. In our case, because each image is already thresholded to black-and-white color, the value of the pixel is 0 for black (the image's background) and 1 for white (the digit's stroke). An illustrative example of this representation is depicted in Fig. 1.

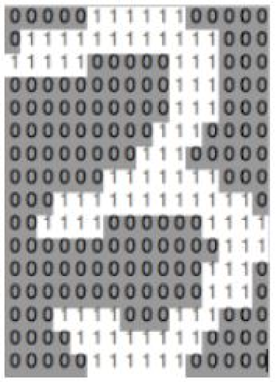


Figure 1: Pixel Representation

3.2 Features

There are many kinds of features that can be extracted from an image. These features may or may not be discriminative between what each image represents. In this section, we consider three function of the PRTools that help us to extract these kind of features.

3.2.1 Image Profiles. The im_profile[2] function of the PRTools[3] computes the horizontal and vertical profiles of an image. A profile is defined to be the normalized image projection of the desired direction (horizontal and vertical). The function receives three inputs: a single image or images dataset A, the number of bins for horizontal profile NX, and the number of bins for vertical profile NY. For this project we pick the number 16 as the number of bins for both the horizontal and vertical profile, thus producing a result of 32 features for each image.

3.2.2 Image Features. The im_features[4] function of the PRTools computes several features related to the image. It uses the regionprops[5] function from Matlab. Some of the features with its short description is explained in Table 1.

Table 1: Image Features from regionprops

Feature Name	Description
Area	Actual number of pixels in the region.
BoundingBox	Smallest rectangle containing the region.
Centroid	Center of mass of the region.
Eccentricity	Eccentricity of the ellipse that has the same second-moments as the region, returned as a scalar.
EquivDiameter	Diameter of a circle with the same area as the region, returned as a scalar.
Euler Number	Number of objects in the region minus the number of holes in those objects, returned as a scalar.
Extent	Ratio of pixels in the region to pixels in the total bounding box, returned as a scalar.
MajorAxisLength	Length (in pixels) of the major axis
Minor Axis Length	Length (in pixels) of the minor axis
Perimeter	Distance around the boundary of the region. returned as a scalar.
Solidity	Proportion of the pixels in the convex hull that are also in the region, returned as a scalar.

3.2.3 Zernike Moments. PRTools also provides the im_moments function to compute several kinds of moments of an image. The moments that we consider to be used for representing the images is the Zernike Moments [4].

3.3 Dissimilarities

Dissimilarity representation as an alternative to the traditional feature representation was introduced in [1]. The idea of this representation is that an object is represented by some similarity measures

[1]https://www.nist.gov/

[2]http://www.37steps.com/prhtml/prtools/im_profile.html
[3]http://prtools.org/
[4]http://www.37steps.com/prhtml/prtools/im_features.html
[5]https://nl.mathworks.com/help/images/ref/regionprops.html

between the object itself and other objects which belong to a set we call as representation set. These similarity measurement become the 'features' of the object to discriminate which class it belongs to.

For example, if we have 100 objects as our representation set, the training set becomes 100x100 matrix which represents the distance between each pair of objects. When we need to classify new object then the measurement between this new object and the objects in the representation set become the 'features' of the new object.

Any kinds of measurement can be used, and in our project we will consider two kinds of measurement: 1-norm distance and city-block distance.

3.3.1 Euclidean Distance. The euclidean distance between two objects q and p is computed using the following formula:

$$\sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

3.3.2 City-Block Distance. The city-block distance between two objects q and p is computed using the following formula:

$$\sum_{i=1}^{n} |q_i - p_i|$$

4 PREPROCESSING

In order to normalize the dataset for building our classifiers, we first apply some basic preprocessing to each image of the dataset. The preprocessing consists of adjusting the bounds of the images and resizing all the image to a uniform size. This section will discuss the detail of each preprocessing step with some samples of the resulting transformation.

4.1 Adjusting Image Boundary

The digit images supplied by the NIST dataset are not uniform in the term of ratio. Some images have longer height, and other have longer width. Some examples of these differences of ratio is depicted in Fig 2.

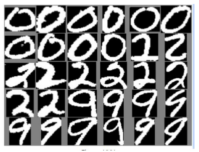


Figure 2: Adjusting Image Boundary

In order to make the dataset to be uniform in term of the ratio between the height and width of the images, we apply the boxing function im_box provided by PRTools. The function can help us to map the images to a square image by adding as many empty columns or rows as possible to the images such that their aspect ratio become 1 (i.e. the width is equal to the height). An illustration of applying this transformation is shown in Fig 3.

4.2 Image Resizing

The second preprocessing that we apply to the dataset is resizing the image such that all images have the same size. In the previous

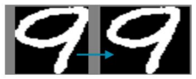


Figure 3: Adjusting Image Boundary

section, we have already determined the ratio between the height and width to be 1, thus we resize the images to a squared size. The original images provided by NIST is 128 x 128 pixels, and to determine which size our images are to be resized to, we experimented with two classifiers: K-nearest Neighbor (knnc) and SVM (svc). We train the classifier on 100 objects (10 objects for each class) with three different image size in pixels: 16x16, 32x32, and 64x64. We evaluate both classifiers using 10-fold cross validation with 10 repetitions. The generalized error rate with its standard deviation is depicted in Table 2.

Table 2: Error rate on different size

Image Size	knnc	svc
16x16	0.377 ± 0.005	0.229 ± 0.022
32x32	0.376 ± 0.005	0.218 ± 0.022
64x64	0.374 ± 0.005	0.209 ± 0.022

As the error rate does not vary too much between these three image size, we decided to resize all of the images to 16x16 pixels, considering the amount of features will be the least among other size choices, thus faster computing time for further experiment. We resize the images using the im_resize function of the PRTools.

5 FEATURE REDUCTION

In order to train our classifier, our dataset especially the one using the pixel representation is very high dimensional. Even for the image size of 16x16, the dimension of the dataset would become 256. This would become troublesome as the computing time would increase as our training set grows, and in the case of Scenario 1 where first we would train with 2000 images, the time taken would be tremendous. This is where we need to introduce feature reduction to our data.

For our digit recognition system, we are going to use Principal Component Analysis (PCA) in order to extract principal features out of our dataset. This section will explain briefly how PCA works.

PCA can be defined as a mapping from high dimension set of measurements to a lower dimension. The main idea of PCA is to find a small number of linear combinations of correlated features to describe as much variance as possible with a small number of new uncorrelated features [3]. The PCA transforms the data to a new coordinate system: the first coordinate (first principle component) captures the greatest variance, the second coordinate captures the second greatest variance, and so on.

The x-th Principle Component (PC_x) is a linear combination of the original features:

$$PC_x = \sum_{i=1}^{N_{tot}} a(i)_x V_i$$

where $-1 \leq a(i)_x \leq 1$ are the coefficients of the linear transformation, V_i are the original features, and N_{tot} is the number of original features.

6 CLASSIFIERS

We have considered 9 different classifiers and finalized on few of them to build our classification system. They are namely K-nearest neighbor, Parzen, logistic, linear discriminant, LDC, QDC, SVM, neural network, logistic classifier. The detailed explanation of its characteristic and its shortcoming are given in this section. We have referred [5] for the explanation of the classifiers.

6.1 K-Nearest Neighbor

K-NN is a Non-Parametric classification method. The 'k' in K-nearest neighbor classifier is the number of nearest objects also known as neighbors that are chosen. The probability density estimation is calculated through k-NN classifier by using adjustable histograms to include k points that are fixed. The volume is large for the density of k points and small for high density of k points. The performance of the estimator degrades in high dimensionality spaces because of insufficient data.

6.2 Parzen Classifier

Parzen Classifier is a Non-Parametric classifier where the dimensional space is divided into hypercubes or windows of length and volume. The probability density function which suffers from the "ancestorâĂŹs sin" are smoothened by kernels also known as potential functions or parzen windows. The influence of each individual Gaussian is more localized in feature space around the mean value.The noise is smoothened out by increasing the values of h. Hence this classifier works best with large data points and smaller bin size.

6.3 Logistic Classifier

Logistic classifier (loglc) is a linear classifier. The log likelihood functions are modeled using linear functions to find the unknown parameters. Optimization is performed based on all the parameters of the unknown probability density function. Logistic classifier relies on lesser assumptions than linear discriminant classifier (ldc).

6.4 Fisher Classifier

Fisher is a linear classifier. This classifier finds the linear discriminant function between classes in the given dataset by minimizing the errors in the least square discrimination. This is a classifier that can be implemented for multiple classes. This uses the one versus others strategy. The output of this classifier is a set of linear classifiers which is assigned one for each class. The final result of this classifier can be improved using a nonlinear classifier.

6.5 Linear Discriminant Classifier

Discriminant functions are used when the class priors are known. Linear discriminant analysis is used when the covariances are equal. The class conditional probabilities are assumed to be Gaussian and the unknown parameters are found by maximizing the log likelihood function. A large dataset is required to find the unknown parameters like mean, class priors and covariance.

6.6 QDC

Quadratic Discriminant Classifier (QDC) is a general form of linear classifier. Here the decision surfaces separating classes are quadratic in nature. The probability density functions are Gaussian distributed.There is no assumption that the covariances of each class are identical. The unknown parameters are found by likelihood ratio.

6.7 SVM

The support vector machine (SVM) is a linear classifier. The goal is to design a hyperplane that classifies the training vectors. The generalization performance of the classifier is used where the classifier designed using the training data set performs satisfactorily with data outside the training set leaving maximum margin for both classes. The goal is to select the direction that gives maximum possible margin. The hyperplanes are determined by a scaling factor. The classes probability density function is minimized subject to the hyperplane. This is a non-linear quadratic optimization task subject to a set of linear equality constraints.

6.8 Neural Network

Neural Networks are non-linear classifiers. Neural networks contain nodes which are categorized as the input, the hidden and the output nodes. The input nodes inputs the data to the system, hidden neurons tries to optimize the weights of the model and the output node combine the outputs of the hidden neurons.

7 EXPERIMENT AND EVALUATION RESULTS

In this section, we will discuss in detail of how our experiment is set up that will lead us to build our final digit recognition system for both Scenario 1 and Scenario 2. First we will explain about the experiment setup, then the estimated performance based on the result of the experiment, and finally the evaluation on the benchmark dataset.

7.1 Experiment Setup

The experiment is held similarly for both Scenario 1 and Scenario 2 with the only difference is the size of the dataset that is used for building the classifier. For each representation that has been discussed in Section 3, we try to build *17 classifiers* including:

- 3 variations of k-nearest neighbor classifier (knnc) with variation of k (1, 3, 5)
- 2 variations of support vector classifier (libsvc) with variation of kernels (linear and quadratic)
- Fisher's linear classifier (fisherc)
- Logistic linear classifier (loglc)
- Linear bayes normal classifier (ldc)
- Nearest mean classifier (nmc)
- Quadratic bayes normal classifier (qdc)
- 3 variations of parzen classifier (parzenc) with variation of h (optimized, 0.25, and 0.3)
- 4 variations of neural network (neurc, rnnc, lmnc, and bpxnc)

Each classifier is trained using the implementations included in PRTools, with some special note for support vector classifier we use libsvc instead of svc as we believe libsvc is faster than svc. The estimated error rate for each classifier is computed using the 10-fold cross-validation technique (prcrossval), repeated 10 times for generalizing the error rate. For the first scenario, only 2000 objects (instead of the whole training set) are used for cross-validation, while for the second scenario the whole 100 objects are used.

For each representation, we choose the best classifiers based on the estimated error rate which is sufficient to reach the target error rate (5% for Scenario 1 and 25% for Scenario 2). Then, we try to introduce feature reduction to each representation using Principal Component Analysis and again estimate the error rate using 10-fold cross validation.

Furthermore, we combine the best classifiers using minimum, maximum, mean, median, and product combiner both using stacked and parallel technique. We compare the estimated error rate between different combiners, again using 10-fold cross validation. Afterwards, we pick the combination technique with the best error rate and build the final classifier using the whole training set (10000 objects for Scenario 1 and 100 objects for Scenario 2). Finally we evaluate the final classifier for both scenarios on the benchmark dataset.

7.2 Estimated Performance

7.2.1 Scenario 1. **Without Feature Reduction:** We have evaluated the performance of various classifiers with 10-fold cross validation method repeated 10 times in order to obtain meaningful statistical number. The performance result of each classifier in each representation is represented in the form of mean and standard deviation of the error rate. The complete result is described in Table 3.

The best performing classifier in each representation can be concluded as follows:

- SVC with degree-2 polynomial kernel for pixel (16x16) representation,
- Loglc for im_feature representation,
- QDC for im_profile representation, and
- SVC with degree-2 polynomial for dissimilarity representation calculated using Euclidean distance.

We decide to discard the Zernike moment representations, be it with or without pre-processing of the images, from further experiment because they perform significantly worse compared to the other representations (more than 60% error rate using all classifier). The results from dissimilarity representation by calculating Euclidean and city-block distance differ insignificantly. To further simplify our experiment, we also pick only one method of distance calculation, that is the Euclidean distance method.

With Feature Reduction: Feature reduction in form of Principal Component Analysis is employed to reduce the number of features in our dataset representations. In order to limit the scope of this experiment, we select the best performing classifier for each representation type to evaluate the performance gained by applying feature reduction to our dataset. To figure out how much variance retention is the most optimal for each classifier, we plot a *graph of error rate* mapping with respect to the number of principal components. The error rates are computed using 10-fold cross validation method. Fig 4, Fig 5, Fig 6, Fig 7 depicts the result of this experiment.

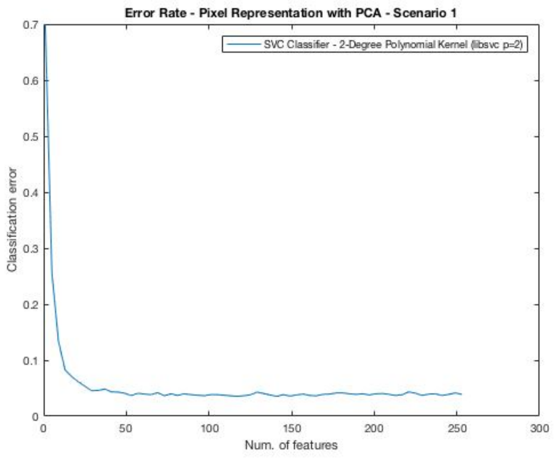


Figure 4: Error Rate - Pixel Representation with PCA - Scenario 1

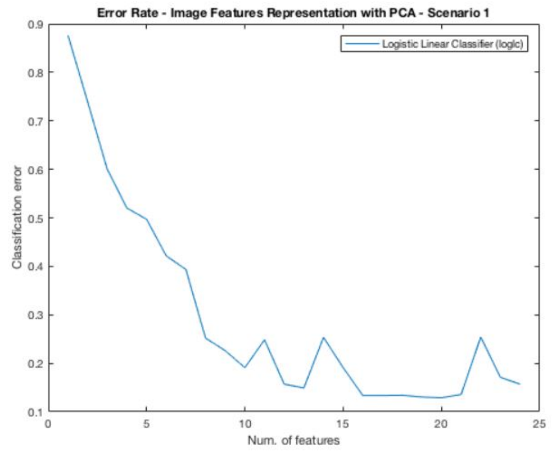


Figure 5: Error Rate - Image Features Representation with PCA - Scenario 1

We choose the number of principal components based on the plot, by heuristically choosing the minimum number that contributes the lowest error rate. The achieved error rate by using the most optimal number of principal components for each representation and its corresponding best performing classifier is depicted in Table 4.

While the number of features for representation drastically decreases, especially with the pixel and dissimilarity representation, the error rate is gaining a slight improvement from the previous result of using all components. Although the gain is relatively not big, we still decide to use the PCA optimized version of each representation as it reduces the dimensionality, thus less time and memory required for training our classifier.

Combined Classifiers: Finally, we combine the classifiers using stacked and parallel mode of combining classifiers. We stack 2

Table 3: Scenario 1 Various Classifiers Performance Result (Error Rate)

Classifier	Pixel	im_features	im_profile	zernike	euclidean	cityblock
knnc (k=1)	0.1147±0.0018	0.4301±0.0046	0.1807±0.0021	0.8411±0.0040	0.1036±0.0025	0.0893±0.0031
knnc (k=3)	0.1268±0.0026	0.4125±0.0064	0.1719±0.0030	0.8438±0.0039	0.1008±0.0021	0.0954±0.0019
knnc (k=5)	0.1275±0.0026	0.4079±0.0041	0.1728±0.0025	0.8328±0.0047	0.0990±0.0024	0.0960±0.0037
svc (linear kernel)	0.0787±0.0025	N/A	0.2377±0.0019	0.7811±0.0039	0.0807±0.0031	0.0844±0.0030
svc (p = 2)	**0.0541±0.0024**	N/A	0.2065 ±0.0021	**0.7662±0.0032**	**0.0739±0.0023**	**0.0788±0.0026**
fisherc	0.1933±0.0036	0.1780±9.8460x10^{-4}	0.2862±0.0032	0.7745±0.0046	0.1052±0.0025	0.1936±0.0035
loglc	0.2459±0.2459	**0.1282±0.0019**	0.2186±0.0024	0.8761±0.0085	0.2133±0.0075	0.2442±0.0051
ldc	0.1521±0.0033	0.1473±0.0014	0.2227±0.0012	0.7776±0.0060	0.1930±0.0023	0.9000±1.1703x10^{-16}
nmc	0.2073±0.0023	0.6367±0.0020	0.3137±0.0023	0.7862±0.0026	0.3669±0.0020	0.3954±0.0016
qdc	0.8351±0.0038	0.1612±0.0081	**0.1633±0.0037**	0.7953±0.0054	0.3222±0.0045	0.8991±8.6442x10^{-04}
parzenc (optimized h)	0.1128±0.0013	0.4212±0.0037	0.1637±0.0035	0.7814±0.0035	0.0959±0.0019	0.9000±1.1703x10^{-16}
parzenc (h=0.25)	0.1133±0.0017	0.5559±0.0023	0.4589±0.0022	0.7780±0.0047	0.4889±0.0019	0.9000±1.1703x10^{-16}
parzenc (h=0.3)	0.1125±0.0021	0.4905±0.0037	0.4713±0.0021	0.7806±0.0042	0.2434±0.0026	0.9000±1.1703x10^{-16}
neurc	0.6611±0.0169	0.1545±0.0026	0.2810±0.0025	0.7602±0.0035	0.6345±0.0043	0.5890±0.0175
rnnc	0.2566±0.0047	0.1701±0.0046	0.2572±0.0041	0.7881±0.0051	0.1957±0.0082	0.2123±0.0054
lmnc	0.5335±0.0348	0.3383±0.0278	0.2958±0.0078	0.7743±0.0071	0.8723±0.0097	0.8740±0.0143
bpxnc	0.7924±0.0269	0.1736±0.0052	0.1940±0.0184	0.7602±0.0085	0.8752±0.0098	0.8792±0.0205

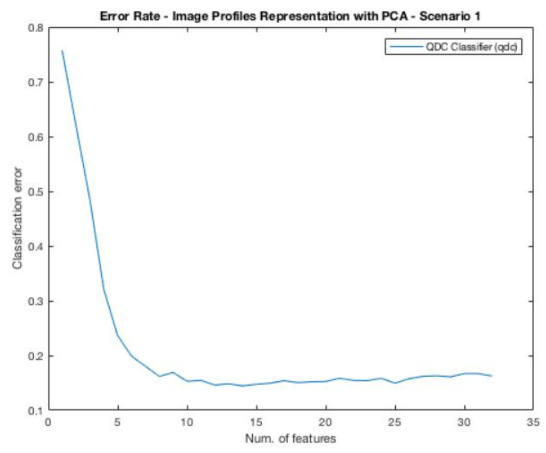


Figure 6: Error Rate - Dissimilarity Representation with PCA - Scenario 1

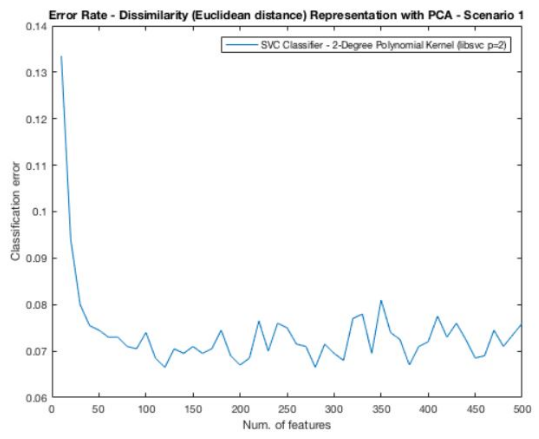


Figure 7: Error Rate - Pixel Representation with PCA - Scenario 1

best performing classifiers on pixel representation, which are svc with p=2 and knnc with k=1. In addition, we also create a parallel mode combination of 4 best performing classifiers for each feature, which are svc (p=2) on pixel representation, loglc on features representation, qdc on profile representation, and finally svc (p=2) on dissimilarity representation. We combine the base classifiers using minimum combiner (minc), maximum combiner (maxc), mean combiner (meanc), median combiner (medianc), and product combiner (prodc). The performance test are done with 10-fold cross validation with 10 repetitions. The result is depicted in Table 5.

As we can see from Table 5, the best estimated error rate is achieved by using parallel median combiner of the four classifiers. Thus, we decide to build the final classifier for the first scenario using this technique.

7.2.2 Scenario 2. **Without Feature Reduction:** The summarized result of 10-fold cross validation for Scenario 2 is summarized in Table 6. The table consists of the error rate and standard deviation of each classifiers for each representation. We picked four best classifiers from the result which are: SVC with linear kernel for pixel representation, LDC for feature representation, 1-NN for profile representation, and Logistic Classifier for dissimilarity representation using euclidean distance. As the error rate for using Zernike moments falls above 60%, we discarded the representation from further experiment. Also we decided to pick the best one of the distance measure for dissimilarity representation, which is the euclidean distance in this case.

Table 4: Scenario 1 - PCA Result

Repr.	Clf	#Feat	#PC	Captured Variance	Error Rate
Pixel	svc (p=2)	256	45	75%	0.0415 ±0.0034
im_features	loglc	24	20	99.99%	0.1300 ±0.0018
im_profile	qdc	32	14	90%	0.1467 ±0.0057
Euclidean	svc (p=2)	500	120	99%	0.0703 ±0.0020

Table 5: Scenario 1 - Combined Classifier Result

Combiner	maxc	minc	meanc	medianc	prodc
Stacked	0.0435 ±0.0007	0.0500 ±0.0014	0.0433 ±0.0004	0.0433 ±0.0004	0.0428 ±0.0011
Parallel	0.1488 ±0.0025	0.0533 ±0.0004	0.1488 ±0.0025	**0.0383 ±0.0025**	0.0445 ±0.0021

With Feature Reduction: For each best classifier of each representation, we try to apply feature reduction using Principal Component Analysis. To determine the number of desired principal components, we plot a graph between the number of principal components and the estimate error rate for each classifier computed using cross-validation. Fig 8 illustrates the plot for each classifier.

We choose the number of principal components based on the plot, by heuristically choosing the minimum number that contributes the lowest error rate. The error rate achieved by using PCA for each classifier and representation is depicted in Table 7,

While the number of features for representation drastically decreases, especially with the pixel and dissimilarity representation, the error rate does not change much from the error rate without using PCA. In fact, there is a small improvement for all of them except for the dissimilarity representation which gain an increased error rate of around 0.8%. Although the change is not much, we still decide to use the PCA optimized version of each representation as it reduces the dimensionality, thus less time and memory required for training our classifier.

Combined Classifiers: Finally, we try to combine the four best classifiers with PCA from the previous section. We combine the base classifiers using minimum combiner (minc), maximum combiner (maxc), mean combiner (meanc), median combiner (medianc), and product combiner (prodc). We do 10-fold cross validation with 10 repetitions for two mode of combining classifiers: stacked and parallel. The estimated error rate and the standard deviation is depicted in Table 8.

As we can see from the table, the best estimated error rate is achieved by using parallel product combiner of the four classifiers. Thus, we decide to build the final classifier for the second scenario using this technique.

7.3 Evaluation on Benchmark Data

As the result of the previous sections, we decided to build the final classifier using parallel median combination for Scenario 1 and parallel product combination for Scenario 2. The classifiers and representations to be combined for Scenario 1 are SVC (p=2) on pixel representation, loglc on features representation, qdc on profile representation, and finally SVC (p=2) on dissimilarity representation. As for Scenario 2, the classifiers to be combined are linear SVC for pixel representation, LDC for features representation, 1-NN for profile representation, and logistic classifier for dissimilarity representation. All the classifiers are trained on the PCA optimized of each representation.

We evaluate our final classifiers on a benchmark dataset using the nist_eval function. We vary the parameter n to 10 and 100 which denotes the desired number of digits per class to be used for the evaluation. The error rate is depicted in Table 9.

8 LIVE TEST

Our trained classifiers from the previous chapter have given us relatively good performance as indicated from both the cross validation and benchmark result. In this chapter, we conduct a live test experiment in which our classifier is tested on our actual handwriting to further evaluate the performance of our classifiers. As the purpose of the experiment is to classify digits written in a bank check, we try to mimic our test image of bank checks to be as similar as possible to the real ones.

8.1 Bank Checks Image Processing

In order to give an accurate resemblance of a bank check, we use a template of colored bank check with light blue background to write our handwriting while the digits themselves are written with a dark blue marker (see Fig 9). Further, we write the digits in two different manners: a good normal handwriting which can be easily distinguishable for a human evaluator and an intendedly bad handwriting which can cause confusion even for a human evaluator. They are done in purpose to gain more insight from our classifier performance. The bank checks are then scanned and the digits are sliced manually into separate bitmap images.

We then process the sliced digits with the following transformation steps:

(1) convert the colored image to grayscale using rgb2gray() function of Matlab. The rgb2gray() function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance,
(2) convert the grayscale image to binary representation using im2bw() function of Matlab. The im2bw() function converts this grayscale image to binary by thresholding. The output binary image BW has values of 1 (white) for all pixels in the input image with luminance greater than the specified level parameter and 0 (black) for all other pixels. We find the best level value of 0.87 from repeated manual observations,
(3) inverse 0 and 1 binary values to mimic the NIST format using imcomplement() function of Matlab,
(4) crop empty spaces besides, above and under the digit by calculating a box of minimum and maximum x and y coordinate

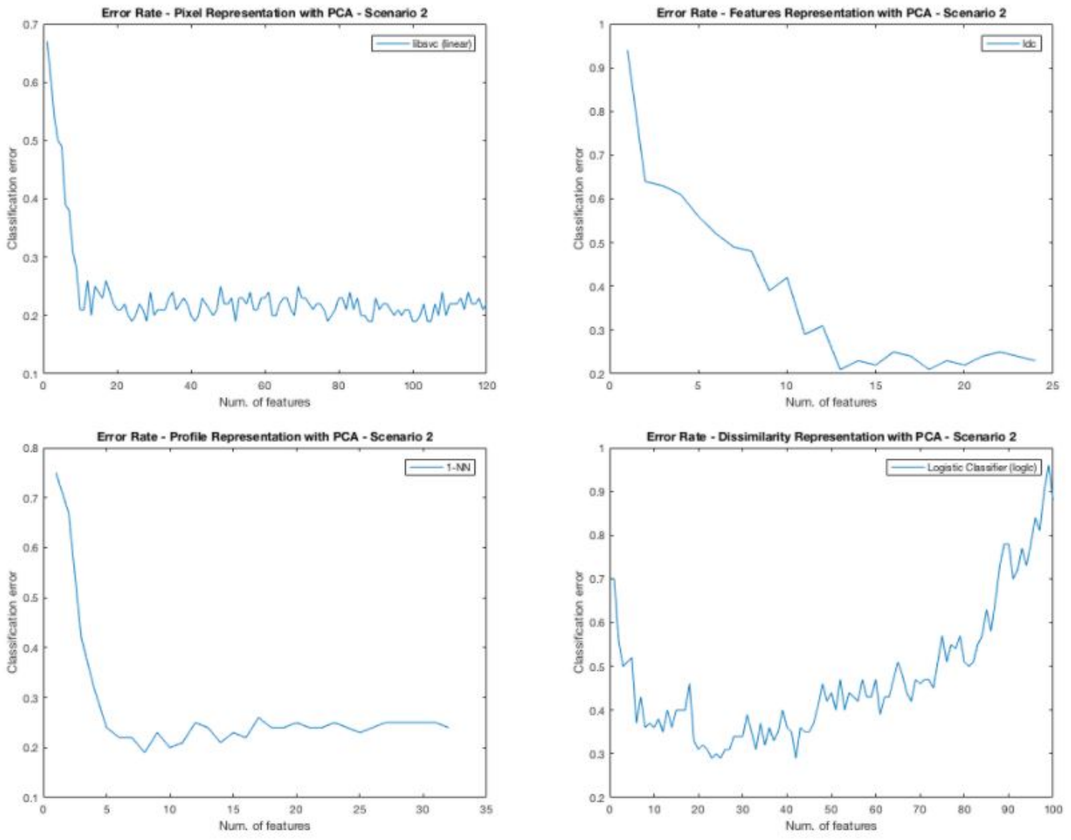


Figure 8: Error Rate - With PCA - Scenario 2

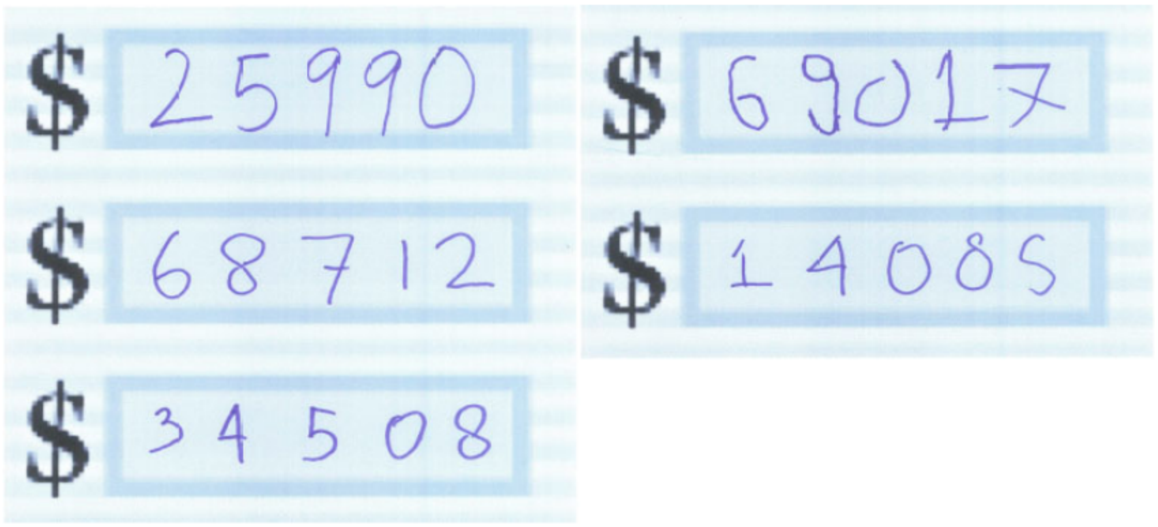


Figure 9: Left: good handwriting; Right: (intendedly) bad handwriting

Table 6: Scenario 2 Various Classifiers Performance Result

Classifier	Pixel	im_features	im_profile	zernike	euclidean	cityblock
1-NN	0.284±0.007	0.5290±0.0057	**0.241±0.009**	0.658±0.009	0.272±0.012	0.391±0.007
3-NN	0.339±0.012	0.6670±0.0189	0.3±0.009	0.698±0.017	0.305±0.009	0.418±0.016
5-NN	0.424±0.0217	0.6800±0.0170	0.334±0.007	0.647±0.012	0.366±0.017	0.382±0.012
svc (linear kernel)	**0.207±0.02**	0.476±0.021	0.453±0.012	0.776±0.014	0.233±0.013	**0.216±0.012**
svc (quadratic kernal)	0.227±0.013	0.514±0.023	0.992±0.006	0.717±0.023	0.252±0.013	0.234±0.011
fisherc	0.382±0.019	0.283±0.0095	0.455±0.021	0.692±0.016	0.215±0.021	0.333±0.022
loglc	0.367±0.0181	0.4390±0.042	0.533±0.017	0.829±0.018	**0.212±0.018**	0.355±0.014
ldc	0.9±0	**0.274±0.0207**	0.394±0.025	0.709±0.014	0.23±0.009	0.264±0.022
nmc	0.285±0.127	0.7180±0.0199	0.277±0.013	0.665±0.014	0.396±0.011	0.364±0.018
qdc	0.9±0	0.5790±0.0256	0.358±0.019	0.7±0.015	0.9±0	0.9±0
parzenc (optimized h)	0.279±0.016	0.6550±0.0190	0.251±0.01	**0.638±0.011**	0.26±0.014	0.398±0.010
parzenc (h=0.25)	0.83±0	0.84±0	0.425±0.012	0.63±0.013	0.88±0	0.9±0
parzenc (h=0.3)	0.77±0	0.82±0	0.422±0.009	0.641±0.011	0.88±0	0.9±0
neurc	0.722±0.0336	0.6840±0.0207	0.691±0.015	0.77±0.036	0.678±0.021	0.692±0.02
rnnc	0.654±0.0417	0.4710±0.0338	0.43±0.045	0.644±0.033	0.56±0.045	0.494±0.039
lmnc	0.762±0.0322	0.6290±0.0515	0.874±0.018	0.817±0.038	0.738±0.06	0.755±0.023
bpxnc	0.868±0.0181	0.8770±0.0221	0.901±0.013	0.893±0.02	0.886±0.021	0.876±0.014

Table 7: Scenario 2 - PCA Result

Repr.	Clf.	#Feat	#PC	Captured Variance	Error Rate
Pixel	svc	256	40	86%	0.201 ±0.02
im_features	ldc	24	15	99.99%	0.225 ±0.013
im_profile	1-NN	32	5	75%	0.23 ±0.009
Euclidean	loglc	100	21	80%	0.298 ±0.025

Table 8: Scenario 2 - Combined Classifier Result

Combiner	minc	maxc	meanc	medianc	prodc
Stacked	0.816 ±0.031	0.545 ±0.041	0.407 ±0.045	0.384 ±0.031	0.404 ±0.037
Parallel	0.218 ±0.016	0.293 ±0.024	0.254 ±0.02	0.243 ±0.019	**0.188 ±0.018**

Table 9: Evaluation on Benchmark Data

	n = 10	n = 100
Scenario 1	2%	2.5%
Scenario 2	19%	23.1%

of white pixel and discard every columns and rows not in that range,

(5) resize the image to 16x16 pixels representation in order to match the image representation in previous section using imresize() function of Matlab,
(6) convert the image into set of doubles, and finally
(7) convert the image set into PR dataset using prdataset() function of PR tools.

The resulting processed digits can be seen in Figure 10.

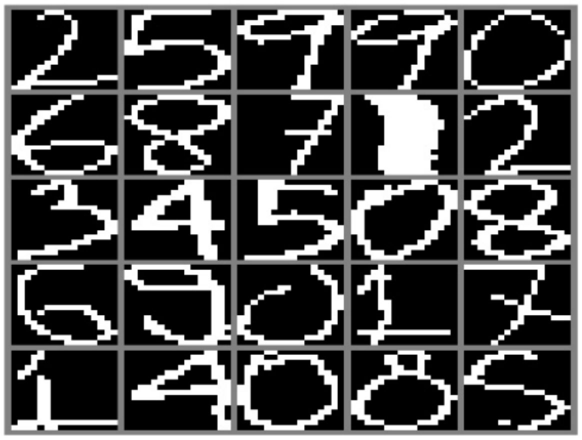


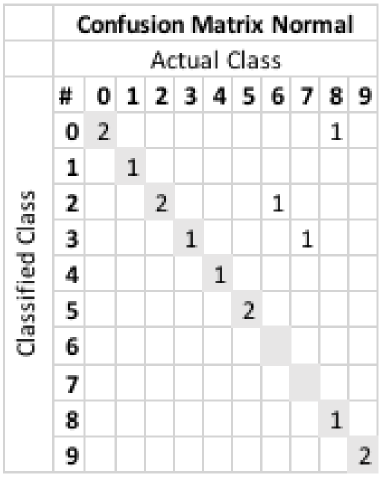
Figure 10: Processed digits

8.2 Classifier Performance

We choose the best performing classifier from previous section, which is parallel mode combination from Scenario 1 using medianc of 4 best performing classifiers for each representation, which are svc with degree-2 polynomial kernel on pixel representation with PCA, loglc on features representation with PCA, qdc on profile representation with PCA, and finally svc (p=2) on dissimilarity representation with PCA. The classifier has been trained with 10,000 images from the NIST dataset.

As we have two separate types of handwriting, i.e. good and 'bad' one, the performance evaluations are also separated into two sub-subsections.

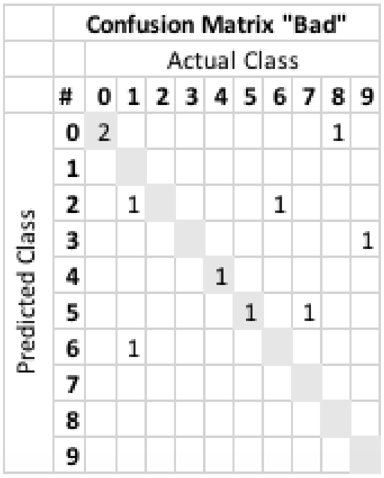
8.2.1 Normal Handwriting. Using the trained parallel combined classifier from Scenario 1, we are able to get error rate of 0.2 from the 15 normally written digits. The confusion matrix is shown on Fig 11.



	Confusion Matrix Normal										
	Actual Class										
Classified Class	#	0	1	2	3	4	5	6	7	8	9
	0	2								1	
	1		1								
	2			2				1			
	3				1				1		
	4					1					
	5						2				
	6										
	7										
	8									1	
	9										2

Figure 11: Normal handwriting confusion matrix

8.2.2 Bad Handwriting. Using the trained parallel combined classifier from Scenario 1, we only get error rate of 0.6 from the 10 'badly' written digits. The confusion matrix is shown on Fig 12.



	Confusion Matrix "Bad"										
		Actual Class									
	#	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	2								1	
	1										
	2		1					1			
	3										1
	4					1					
	5						1		1		
	6		1								
	7										
	8										
	9										

Figure 12: Bad handwriting confusion matrix

The not-so-satisfying result in this sub-experiment is not surprising as the digits are not easily distinguishable, even for a human observer.

9 CONCLUSION

As conclusion, a large dataset is important during the development of a classifier system. As per guidelines provided in [5], we can achieve the minimum error that is the Bayes error. But that is not always the case we see in Scenario 2 used for training and testing the classifier we have built. We have chosen three kinds of representation: pixel representation sized to 16x16 pixels, features representation including image profiles, image features, and zernike moments, and dissimilarity representation.

We have experimented with 17 classifiers which includes their variations and a combination of those classifiers that is mentioned in detailed manner in Experimental Setup section. We have especially used libsvc function for building classifier system as it is faster and prcrossval function for cross-validation and repeated it 10 times to generalize error rate. We have used 2000 objects for scenario 1 while for the second scenario the whole 100 objects are used for cross-validation. At the end we have chosen a combination of classifiers and built the final classifier using the whole training set i.e. 10000 objects for Scenario 1 and 100 objects for Scenario 2. The performance of the classifiers are represented statistically by mean and standard deviation of the error rate. The *best performing classifiers* that we have finalized are : SVC with degree-2 polynomial kernel for pixel (16x16) representation, Loglc for im_feature representation, QDC for im_profile representation and SVC with degree-2 polynomial for dissimilarity representation calculated using Euclidean distance.

For **Scenario 1**, we have achieved less error rate by building the final classifier using *parallel median combiner* of the four best performing classifiers for each feature, which are svc (p=2) on pixel representation, loglc on features representation, qdc on profile representation, and finally svc (p=2) on dissimilarity representation. The performance result is displayed in Table 5 and specifically parallel median combiner gives the least error rate of **0.0383 ± 0.0025** .

For **Scenario 2**, we have achieved less error rate by building the final classifier using *parallel product combiner* of the four best performing classifiers for each feature: SVC with linear kernel for pixel representation, LDC for feature representation, 1-NN for profile representation, and Logistic Classifier for dissimilarity representation using euclidean distance and specifically parallel product combiner gives the least error rate of **0.188 ± 0.018**.

Finally, we tested our final classifiers for scenario 1 and 2 using the nist_eval function and vary the parameter 'n' to 10 and 100 and the result is shown in Table 9 and the result is 2% and 2.5% for scenario 1 and 19% and 23.1% for scenario 2.

10 RECOMMENDATIONS

We are aware that there are many improvements that can be introduced in order for the digit classification system to perform better. In this section we will discuss some of potential points that can be improved in the future.

The first point is to introduce rejection option to our system. The cost of misclassification of monetary amount or cheque numbers can cause a serious problems with money transfer regarding wrong account number and wrong amount to be transfered. Hence we can *reject* the misclassified digits in the future and it will increase the overall performance of the classifier.

Also, in the preprocessing step we can introduce more sophisticated process such as slant correction as described in [6]. The purpose of slant correction is to fix images of digit which is skewed. Other preprocessing step that can also be included is the closing operation to fix gaps between digit strokes caused by low quality image from scanning the handwriting.

In our system, we used three kinds of digit representation: pixel, features, and dissimilarity. Another popular feature that we didn't use for digit recognition is Histogram Oriented Gradient (HOG) feature. HOG is derived by counting the occurrences of gradient direction in several portions of the digit image. It is proven to be robust for digit classification as described in [2].

Finally, for a more practical use, online learning can be incorporated to the classifier system. As opposed to offline learning where we train our classifier on a fixed number of dataset and use the classifier for the system, we can keep improving the classifier by retraining it as more dataset is available. This is especially useful for the case such as scenario 2 where the number of initial training set is not large.

REFERENCES

[1] Robert Duin and Elżbieta PeÌğkalska. 2011. The dissimilarity representation for structural pattern recognition. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (2011), 1–24.

[2] Reza Ebrahimzadeh and Mahdi Jampour. 2014. Efficient handwritten digit recognition based on histogram of oriented gradients and svm. *International Journal of Computer Applications* 104, 9 (2014).

[3] M Einasto, LJ Liivamägi, E Saar, J Einasto, E Tempel, E Tago, and VJ Martínez. 2011. SDSS DR7 superclusters-Principal component analysis. *Astronomy & Astrophysics* 535 (2011), A36.

[4] Alireza Khotanzad and Yaw Hua Hong. 1990. Invariant image recognition by Zernike moments. *IEEE Transactions on pattern analysis and machine intelligence* 12, 5 (1990), 489–497.

[5] Sergios Theodoridis and Konstantinos Koutroumbas. 2010. Pattern Recognition & Matlab Intro. (2010).

[6] Takuma Yamaguchi, Minoru Maruyama, Hidetoshi Miyao, and Yasuaki Nakano. 2005. Digit recognition in a natural scene with skew and slant normalization. *International Journal of Document Analysis and Recognition (IJDAR)* 7, 2-3 (2005), 168–177.