IN 4085 Pattern Recognition
Hand Written Digit Classification
Group 70

Dimitropoulos Georgios: 4727657
Manousogiannis Emmanouil: 4727517
Tsoni Sofia: 4747941

February 2, 2018

1 Introduction

Automatically recognizing handwritten characters is a main research field within pattern recognition. There is a vast research area covering the recognition of handwritten digits and characters [1] ,[2]. A wide spread application of this field is its utilization for the recognition of monetary amounts for automatic bank cheque purposes. A significant amount of research is conducted using the NIST dataset [3] ,[4]. The objective of this paper is to demonstrate the use of different classifiers in order to decrease the classification error in the automate classification of individual digits used for cheque purposes. More specifically, the use of K Nearest Neighbors, Parzen, logistic, Fisher, Linear Discriminant, Nearest Mean, Quadratic Discriminant,as well as advanced classifiers such as Neural Networks and Support Vector Machines and combination between classifiers is investigated for two different scenarios. In both scenarios, each digit represents each one of the ten classes since digits from 0 to 9 are classified. The first scenario includes a training set of at least 200 out of 1000 digits per class and the second scenario a maximum of 10 digits per class. Moreover, preprocessing consists a necessary step before training or classification of the data. The three basic preprocessing steps which we follow in our approach, after converting our images into binary form, include : Dilation and erosion, central moments and resizing. Furthermore, different representations for our input dataset are used, including pixel, feature and dissimilarity representation. Overall a classification error in an independent benchmark of 3.5% for scenario 1 and 9% for scenario 2 is achieved with, the use of the pr tools function nist_eval. These results are attained by the Parzen classifier through the pixel representation of our data. Moreover,these classifiers are further evaluated using a live test set of the author's handwritten digits with classification errors of 25% and 65 % for scenario 1 and scenario 2 respectively.

The rest of the paper is organized as follows: Section 2 refers to the preprocessing steps which are performed before the training of our classifiers. Section 3 presents the ways used in order to represent our data. Namely pixel, feature and dissimilarity representation have been adopted. Section 4 provides information about the Principal Component Analysis (PCA) method, whereas Section 5 analyzes the basic characteristic of our proposed classifiers. Section 6 presents the experiments which we conducted in our work for all classifiers for each of the three types of representations of our data and for both scenarios. In Section 7, the benchmark experimental analysis is performed. In section 8 a live test is conducted in order to apply the proposed classifiers which achieve the highest accuracy to a specific domain of digit recognition. Section 9 summarizes our work, where interesting conclusion could be depicted. Section 10 includes recommendations and future work. Finally, references of suggested bibliography which has been studied in depth are mentioned.

2 Preprocessing

Preprocessing consists a necessary step before training or classification of the data. Initially the data that was provided consisted of a set of images of different sizes which contain handwritten digits.The three basic preprocessing steps that we follow in our approach, after converting our images into binary form, include: dilation and erosion which are used for noise reduction.The use of central moments is also adopted for orientation and slant correction. Finally, scaling is used through resizing to 30 by 30 pixels our images, as this will be a great boost to the performance of many of our scale sensitive classifiers (i.e kNN or Parzen).In total, these preprocessing steps are used in order to filter out noise and transform images in a way that not only structure of the images is maintained but also the data is prepared for feature selection, extraction and classification.

2.1 Noise Reduction

As far as the reduction in noise is being considered,a technique that takes the mathematical morphology of the image into consideration was chosen. This technique is about the analysis and processing of geometrical structures, based on set theory, lattice theory and topology.

More precisely erosion - dilation are used, which are the two basic morphological operations. Noise is reduced by first eroding and then dilating the given binary input image using a 2-dimensional disk shaped kernel. In our case, the boundaries of regions of the foreground pixels are eroded away during the erosion operation on a binary image, as we have gray-scale morphology. This leads to the shirking of the foreground pixels. Thus, holes within those areas become larger. After that the dilation of the image follows. The basic effect of this operator is to gradually enlarge the boundaries of regions of foreground pixels, which is the exact opposite procedure to erosion. The dilation uses the aforementioned structuring element in order to expand the shapes contained in the input image. Disk case was adopted as choosing interesting information about the digits, lies in the central cross of each image. Demonstration of the techniques of erosion and dilation can be depicted in figures 1 and 2, before and after their implementation on images respectively.

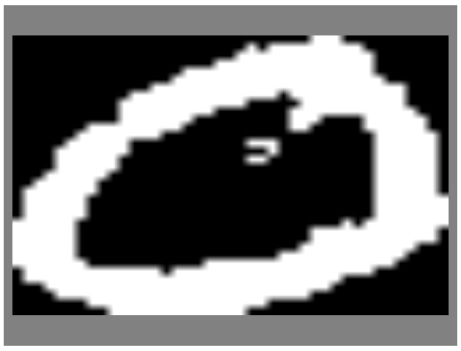
Figure 1: Digit before erosion and dilation

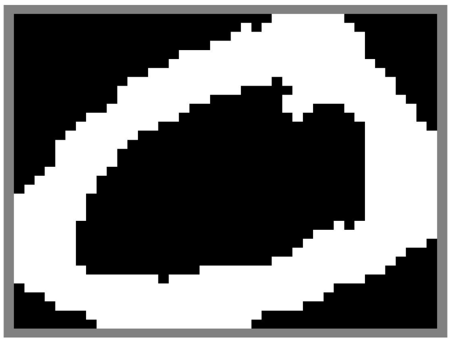
Figure 2: Digit after erosion and dilation

2.2 Slant Correction

Some times digits are written crooked. This does not help us to the classification problem, because we are expecting the main information of the image to be in the x, y axis, namely in the center of the image. Through observations, it is noticed that most of the digits contain more information in the y axis. Namely it is centered and oriented in such a way that the longest side of the number is parallel to the y axis. The transformation which is used, skews and removes the slanting from the text. Hence, the use of moments is adopted. The main advantage of geometric moments is, that they provide rich information about the image. Moments due to their highly informative content can be used as an equivalent representation of an image, in the sense that an image can be reconstructed from its moments. Thus, each moment's coefficient conveys a certain amount of the information residing in an image [5].

This operation except for relocating the image in the correct degree, also decreases the variance of the pixels in the corners as these pixels will be activated rarely in comparison to other, and they contribute less to this hand written digit classification system [6].

The methodology we adopt is as follows: Firstly, the center of the image is calculated. More specifically we find the image centroid, which is the mean of the x and the y axis:

$$\overline{x} = \frac{(x_1 + x_2 + x_3 + ... + x_n)}{n_x} \tag{1}$$

$$\overline{y} = \frac{(y_1 + y_2 + y_3 + ... + y_n)}{n_y} \tag{2}$$

In order to find the orientation of the binary numbers, the central moments have been also adopted. Central moments are calculated using the following formula:

$$\mu_{ab} = \sum_x \sum_y (x - \overline{x})^a (y - \overline{y})^b \mathsf{f}(x, y) \tag{3}$$

Next we use this definition to compute $\mu_{11}, \mu_{20}, \mu_{02}$ which represents the x and y variance and covariance respectively. After that, these central moments are used in order to estimate the orientation.

$$orientation = \frac{1}{2} \arctan(\frac{2 \times \mu_{11}}{\mu_{20} - \mu_{02}}) \tag{4}$$

Last but not least, in order to preserve the geometric information, the image is skewed in parallel to the y axis and through this, we find the bounding box and square it where the sides are equal to the longest previous side. The slant correction skew matrix is given by:

$$S = \begin{bmatrix} 1 & 0 \\ \sin(0.5\pi - \theta) & \cos(0.5\pi - \theta) \end{bmatrix} \tag{5}$$

Demonstration of a digit before and after the implementation of the slant correction technique can be depicted in figures 3 and 4 respectively.

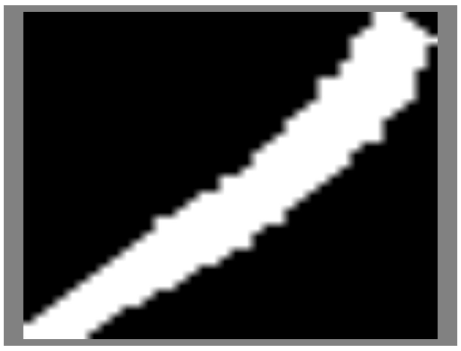
Figure 3: Digit before slant correction

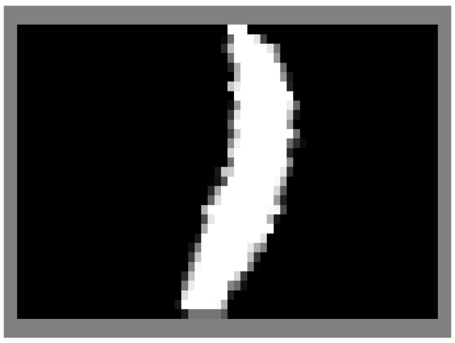
Figure 4: Digit after slant correction

2.3 Resize of the Image

Scaling of an image is used When we need to create an image with different dimensions from the given. In the provided dataset, not all images have the same size and thus they had to be resized in order to avoid poor performance of our scale sensitive classifiers . Through experimentation we decided to resize the images to 30 by 30 pixels.

3 Representations of the data

The next section analyzes the representation of our image dataset either with pixel, feature or dissimilarities representation [7].

3.1 Pixel Representation

The pixel representation, in its broadest sense, samples the objects and uses them to build a vector space. It is one of the simplest representations to define and is based on the simple neighborhood relations between elements. One drawback of the pixel representation is that it does not take advantage of the spatial connectivity of the pixels in the images.

The main idea behind pixel representation is the fact that one set of pixels, representing one specific handwritten digit image, that is almost identical with another set of pixels of another image, corresponds to the same class.

3.2 Feature Representation

In feature representation technique, each image of our dataset is mapped and represented by a set of features, which are predefined in *imfeatures* function of *PRTools*. Examples of those features are 'Area', 'Orientation' and 'Perimeter' of each image object. The goal of feature representation is to map our images into another domain in which they can be compared computationally. Usually, this domain is a vector space. In an accurate feature representation technique, small differences between represented images, correspond to similar objects.

Features are well suited to represent objects in a numerical way, but their position should be known and this knowledge can be presented in measurable quantities. The second problem of the feature representation is that human knowledge is not naturally specified in measurable properties.

3.3 Dissimilarities Representation

Apart from the above two representations which represent objects in numerical ways, dissimilarity measures have been used for the representation of the data. In such a case, if d_{ab} is a distance metric between two objects a and b, then if $d_{a,b} < \delta$ and δ is quite small,then we can assume that we deal with objects of the same class. Dissimilarities between objects are measured directly on the raw data and not on the basis of measured properties. The advantage over the feature representation is that every aspect of the object can be taken into account.

In our case three different types of distances, namely the Euclidean distance, the city block and the Minkowski distance have been used. All of them had been evaluated and the one that led to the highest accuracy had been adopted for the representation in Scenario 1 and 2.

4 PCA for Feature Extraction

Principal Component Analysis (PCA) [8] is a data extraction technique that generates a set of principal components such that as much variance as possible between the features is maintained. The rational behind this, is that through PCA, the feature set becomes smaller and as consequence the classifier becomes faster. The mathematical background behind PCA is as follows:

Let X be a feature matrix with its rows representing different samples and the columns representing different features. We first subtract the mean of each feature from the columns so that the mean of each column, becomes 0 in order to make a new matrix called Y. Then an eigenvalue decomposition of the covariance of this matrix is performed namely $PDP1 = cov(Y)$. By sorting the eigenvectors based on the corresponding eigenvalues, the L vectors that contribute most to the data can be found. The highest eigenvalues correspond to the eigenvectors that capture most covariance. The multiplication of our data with these eigenvectors $W = P_L Y$ will lead to the reduced feature set. In our approach, the *PRTools* function *pcam* in order to perform the above feature extraction is used. In addition, the *prtools* function *featself* that uses the above method, further reduces the feature vectors as described below.

As mentioned,the *featself* command by the *PRTools* library has been used for further feature reduction. This function, in order to choose the most suitable features, receives as an input the measure in which the distance between the features is based on. For this criterion, there are six considerable options: inter-intra distance, sum of estimated Mahalanobis distances, minimum of estimated Mahalanobis distances, sum of squared Euclidean distances, minimum of squared Euclidean distances and 1-Nearest Neighbour leave-one-out. In order to find the most suitable one, the function *feateval* was used, and each time the code is executed, the distance with the higher evaluation value is chosen. Through experimentation we realized that the most frequently used was the sum of estimated Mahalanobis distance.

As far as the feature selection strategy is being considered, more features are kept for scenario 1 and less for scenario 2. This decision was based on the fact that larger data sets require more features for a most informative representation of the data.

5 Proposed Classifiers

A significant number of classifiers which are proposed in [9] are studied in this paper.

5.1 k Nearest Neighbors

K-NN is a non parametric classifier, which uses k-NN density as an estimation of the classification of the data points. k is the number of the neighbors, which needs to be optimized using either cross-validation or other optimization techniques.Based, on that we used 1-NN ,2-NN and 3-NN classifiers.

5.2 Parzen

Parzen classifier is a non-parametric classifier, which uses Parzen density estimation via using kernels for smoothing in combination with Bayes rule, for classification tasks.Before using Parzen classifier, we tried to optimize h parameter is the kernel size,using an optimization technique that gives the highest log-likelihood error.

5.3 Logistic

Logistic classifier is a linear classifier, which uses linear equations for the determination of the decision boundaries.

5.4 Fisher

Fisher is a linear classifier, which tries to minimize the least square error.

5.5 Linear Discriminant

Linear Discriminant (LDC) is a Bayes plug-in classifier which makes the assumption that all classes have the same covariance matrix.

5.6 Nearest Mean

Nearest Mean (NM) classifier, classifies each object based on the euclidean distance between the object that is going to be classified and its mean from each class. It makes the assumption that the covariance matrix is identical.

5.7 Quadratic Discriminant

Quadratic Discriminant (QDC) is a Bayes plug-in classifier which is based on the assumption that the means and the covariance matrix are different for each of the classes.

5.8 Neural Networks

Neural networks (NN) are non-linear classifiers, which were inspired by human brains. Neural networks contain nodes (neurons), which are categorized in three categories, namely the input, hidden and the output neurons. The input neurons feed data to the model, whereas hidden neurons are used to optimize the weights of the system and finally the output nodes combine the outputs of the hidden neurons.

5.9 Support Vector Machines

Support Vector Machines (SVMs) are a linear classifier, which tries to find a boundary that maximizes the margin between the support vectors.

6 Experimental Results

In order to be able to perform our experiments, we based on the PR tool-box for *MATLAB* [10]. For the evaluation of our system, three different error estimation procedures are used . We calculated the Hold Out error, implemented 5-fold Cross Validation and 10-fold Cross Validation. In order to estimate the error of our system we decide how large the train set and the test set will be. A common compromise for the above dilemma is a 50-50 split of the design set in equal parts for training and testing. Using this kind of split we calculated the hold out error, which is usually slightly pessimistic. As far as the cross validation is being concerned, n classifiers are trained by a fraction of $\frac{n-1}{n}$ of the design set and tested by the remaining objects, $\frac{1}{n}$ of the design set.

6.1 Scenario 1

The Scenario 1 has been executed for all three types of representations, pixel, feature and dissimilarities. For this case, more than 200 objects per class were needed and thus we tested our system with 200 and 500 objects per class. The best results were achieved with 500 objects per class so this scenario was eventually preferred.Even if we in further increased the number of objects per class, the results were not actually improved, so there was no actual reason to obtain more.

For each representation we had to approach the dimensionality reduction of the representation space differently. Pixel representation, as mentioned above, has the biggest space of all the other representations. Thus, in our code the dimensions of the pixel representation are always higher than the others. In feature representation we have a feature space of size 20. That means that the feature space is already small enough and we have to be cautious to any other reductions.

In the following figures, it is shown how the error changes as the number of features increases (figures 5-13). These graphical illustrations helped us choose the best number of features for the classifiers. We plotted the corresponding graphs for the other representations and we chose the number of features where the error seems to be the lowest (table 1). The fact that after one point, the error starts to increase instead of decrease is caused due to overfitting. Overfitting occurs when a model learns the detail (even the noise) in the training data, to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data are picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

Representation	Selected Features
Pixel	30
Features	20
Dissimilarities	30

Table 1: Number of features selected for each representation

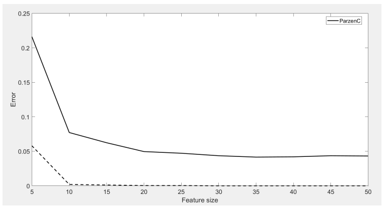

Figure 5: ParzenC classifier

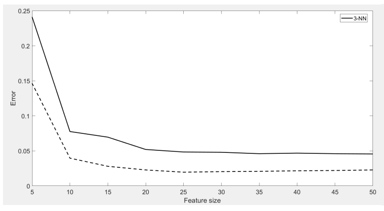

Figure 6: 3-NN classifier

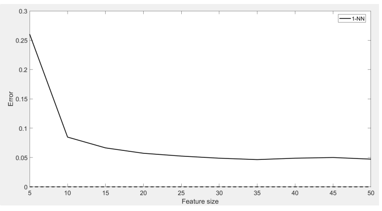

Figure 7: 1-NN classifier

As can be seen in tables 2 and 3, better results were achieved when using pixel representation, compared to feature or dissimilarities representation. However,based on theory, dissimilarities representation seems to generally achieve the highest performance, while feature representation should also outperform pixel.We managed to obtain remarkable results with pixel representation of our data, probably because of the quite good preprocessing of the dataset and the PCA feature extraction technique. Since, as described in section 7, we managed to achieve the requested benchmark in this way , we did not insist further on improving our relatively poor results in the other two representation methods. However, we could possibly improve our results in many ways. For instance,

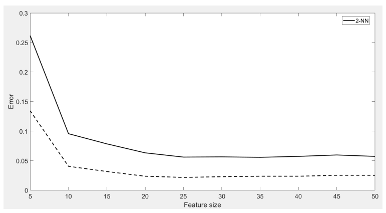

Figure 8: 2-NN classifier

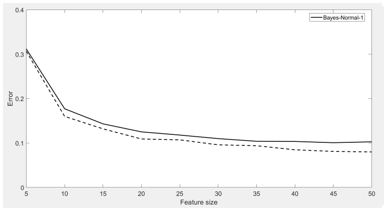

Figure 9: Bayes Normal 1 classifier

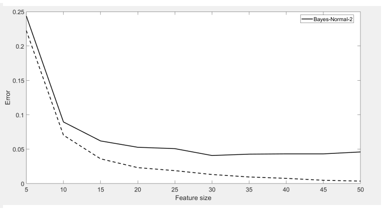

Figure 10: Bayes Normal 2 classifier

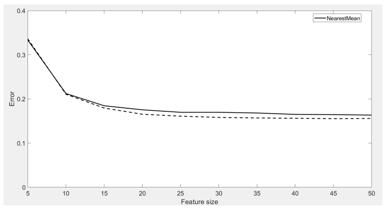

Figure 11: Nearest Mean classifier

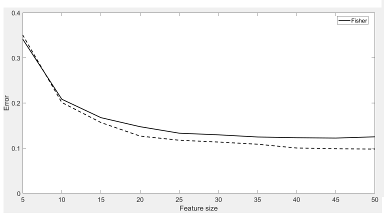

Figure 12: Fisher classifier

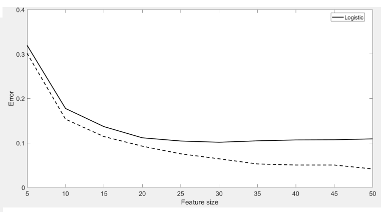

Figure 13: Logistic Discr. classifier

Classifier	**Pixel Repr.**			**Feature Repr.**			**Dissimilarities Repr.**		
	HO	Cross5	Cross10	HO	Cross5	Cross10	HO	Cross5	Cross10
1-NN	4.4%	5.4%	5.3%	35.4%	35%	34.2%	27.2%	27%	25.8%
2-NN	5.9%	7.3%	6.2%	36.9%	37.2%	35.9%	30.2%	30.9%	30.2%
3-NN	4.8%	6.3%	5.5%	32.6%	32.2%	32.5%	30.4%	31.1%	31.3%
Nearest Mean	15.8%	16.6%	16.2%	53.9%	58.8%	59.1%	44.3%	45.8%	46%
Linear discriminant	10%	11.6%	11%	13.2%	14.2%	14.8%	11.9%	11.2%	10.8%
Quadratic Discriminant	3.8%	5.3%	4.4%	13.6%	12.9%	13.3%	14.2%	13.7%	14.1%
Fisher	12.3%	12.9%	6.3%	18.4%	18.2%	18.8%	19.2%	18.9%	19.1%
Logistic	9%	11.5%	5.1%	27.5%	26%	26.5%	25.4%	25.8%	26%
Parzen	4%	5%	4.2%	24.7%	24.2%	24.5%	22.4%	22.6%	22.5%

Table 2: Error Rates for Scenario 1, for all representation techniques

Advanced Classifier	**Pixel Repr.**			**Feature Repr.**			**Dissimilarities Repr.**		
	HO	Cross5	Cross10	HO	Cross5	Cross10	HO	Cross5	Cross10
Support Vector Machine	9.1%	10.9%	10.1%	13%	13.9%	13.6%	12.3%	11.9%	11.4%
Linear Perceptron	12.7%	14%	13.2%	24.2%	22.7%	22.1%	23.3%	22.7%	22.4%
Voted Perceptron	9.3%	9.5%	9.1%	35.7%	34.6%	34.9%	39.2%	38.6%	38.1%
DRBM	13%	13.3%	13.8%	42.9%	43.1%	43%	39.8%	38.5%	38%

Table 3: Error Rates for Scenario 1 using advanced classifiers, for all feature representation techniques

in feature representation , we could use only part of the available features provided, as the total number of them may cause over fitting or/and be uninformative. Additionally, we could perform Fisher mapping techniques, instead of PCA , for feature extraction as it is preferable technique in terms of class separability.

6.2 Scenario 2

The Scenario 2 has been executed for all three types of representations, pixel, feature and dissimilarities. For this case, a small dataset was required in order to train and test the classifiers. The design set used contained 10 objects per class.

Quadratic Classifier, which was one of the best for the larger dataset in scenario 1, here had the worst performance with big difference. This occurs due to the complexity of this classifier. In this scenario we have too few data for a non linear classifier to fit properly. Nearest Mean, which had

Classifier	**Pixel Repr.**			**Feature Repr.**			**Dissimilarities Repr.**		
	HO	Cross5	Cross10	HO	Cross5	Cross10	HO	Cross5	Cross10
1-NN	17.5%	32.1%	29.3%	52%	57.2%	51.2%	49%	52.7%	50.3%
2-NN	25%	35.6%	39.8%	54%	57.5%	56.8%	47.2%	53.5%	52.8%
3-NN	27.5%	37.1%	45.5%	58%	69%	66.8%	52.4%	59.2%	56.6%
Nearest Mean	27.5%	24.1%	34.8%	48%	64.2%	61.2%	47.8%	53.1%	50.9%
Linear discriminant	37.5%	47%	25%	25.8%	24.3%	31.7%	22.9%	20.7%	20.2%
Quadratic Discriminant	87.5%	89%	69.1%	68%	66.2%	64.2%	68%	66.2%	64.2%
Fisher	20.5%	26.5%	25.1%	24%	41.2%	34.7%	28.2%	37.5%	35.4%
Logistic	30%	49%	49.3%	54%	68.7%	53.2%	51.2%	54.6%	49.1%
Parzen	17.5%	25.8%	26%	44%	63.7%	50.3%	41.2%	52.2%	48.9%

Table 4: Error Rates for Scenario 2,using only 10 objects per class, for all feature representation techniques

Advanced Classifier	**Pixel Repr.**			**Feature Repr.**			**Dissimilarities Repr.**		
	HO	Cross5	Cross10	HO	Cross5	Cross10	HO	Cross5	Cross10
Support Vector Machine	35%	47.9%	50.1%	59.2%	60.5%	62.2%	60.1%	59.8%	59.5%
Linear Perceptron	32.5%	47.5%	51.2%	40%	41.7%	45.8%	38%	37.9%	37.2%
Voted Perceptron	35%	54.5%	57.7%	76%	68.2%	71.5%	73.2%	70.9%	70.4%
DRBM	37.5%	47.2%	55.2%	54%	64.2%	61.5%	52%	58.9%	57.2%

Table 5: Error Rates for Scenario 2 using advanced classifiers, for all feature representation techniques

a really bad performance in scenario 1, in this case had much better performance in comparison with the others. Generally complex classifiers here perform worse than the more simple ones. This happens because we had just 10 objects per class and approximately 30 pixels per case.

7 Benchmark

In this section, the goal is to assess our system with data withheld by the client. From the previous sections we have made our decision about which representation provides the best results and also which classifiers seem to be the best for our case.Consequently, only these options are used for the evaluation of the system. For our information we will provide a comparison between the classifiers that had the best accuracy in the previous sections, and not just one.

For the scenario 1, we compared Parzen and Quadratic Discriminant which provided the best results in the experiments above. Using the benchmark, Parzen had the best performance and is the classifier which we will propose to the client. Additionally we see that Parzen had the best results for the scenario 2 too. This seems to be unexpected based on theory, since Parzen performs better for bigger train set. For scenario 2 the second best classifier is Fisher, with a higher error rate than Parzen.

Classifier	Error Rate
Parzen	3.5%
Quadratic Discriminant	4.1%

Table 6: Benchmark Results for Scenario 1 using Pixel representation

Classifier	Error Rate
Parzen	9%
Fisher	12.3%

Table 7: Benchmark Results for Scenario 2 using Pixel representation

8 Live Test

The final test of our system was performed examining the live test operation. Our handwritten digits were scanned and used for this purpose. Handwritten digits that are used for this purpose can be depicted in figure 14. Some digits were made on purpose more ambiguous (a few 1 and 7 are similar) as it was desirable to test how well it performs on occasions close to reality.

The first step towards this was a kind of preprocessing. Namely, separation of all digits of the scanned image into individual digits was performed. Adjustments, like inverting its color, were done in this step. The next step included the addition of the individual images into an array, followed by converting them into prdataset form. After that, the classifier should be trained. For this purpose NIST dataset was used. Finally the performance of different classifiers trained by NIST dataset was evaluated on our own handwritten digits. Through experimentation it was found that for both scenario 1 and scenario 2 the best performance is obtained for pixel representation and for the Parzen classifier. Specifically, for scenario 1 error rate of 25% and for scenario 2 an error rate of 65% are achieved.

The error rate is much higher than the one when evaluating our system with the *nisteval* function. There are many reasons that may degrade the performance of our system when testing with our handwritten digits. For example errors in the process of construction of the individual digit images can be a reason of this phenomenon.

Finally, it was evident through experiments that classifiers had a difficulty in recognizing 7's and 1's in some cases. Hence, it could be depicted that building a generic digit recognition system which should be able to classify digits from a variety of TU Delft students is a much more challenging problem with different aspects and requirements. Last but not least, really crucial is the decision of the dataset that we will choose in order to train our system.

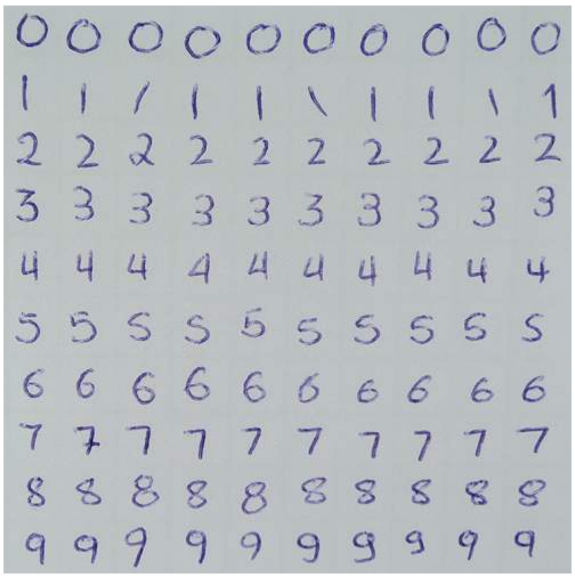

Figure 14: The handwritten digits for the live test

9 Conclusion

To conclude, finding an optimal solution for the problem of the digit recognition consists a multidimensional issue which depends on various parameters. Also, it is evident that there is not an ideal ("best") classifier. Within the experiments the images are resized to 30X30 pixels. This choice was done in order to achieve the perfect matching between the runtime of classifiers and the runtime of calculating the dissimilarity matrices, in combination with a possible impact on the classification error, especially when pixel representation is adopted.

Cross-validation in the experiments is used to separate the training and test data. In both cases 5-fold and 10-fold cross-validation with 10 repetitions are used. Also the 50-50 split case is studied. Although cross-validation results in less over fitting some over fitting can still be depicted.

Through experimentation it is noticed that, in general, simple classifiers (i.e. k-NN and Parzen) perform better in this specific problem than complex classifiers (i.e. neural networks and SVMs) as the latter ones require optimization. Noticeably, in scenario 1 and 2 some classifiers completely collapse and result in tremendous errors.

Feature reduction is a way than can lead to a significant improvement of the classification error.By implementing the PCA technique method, the classification error is reduced. It is worth mentioning that the number of principal components in pixel representation is on average reduced from 900 features to around 30 principal components in both scenarios.

Overall, the classification error as determined by the cross-validation method, differs from the results obtained by nist eval. In scenario 1, the difference is slightly small (around 0.5%) and is possibly related to a bias in the training and test set. Also, nist eval tests the trained classifier just once with one test set derived from the initial test data. Hence there in a variance in the obtained results. Regarding scenario 2, the difference between the cross-validation result and the benchmark result is comparably high (around 8%). When a classifier is trained on a small dataset, there is a bias towards this particular dataset. Within cross validation, the training and the test data are drawn from one dataset. We made the assumption that this represents a dataset close to the true distribution. However, in case nist eval tests the classifier with a completely different and random dataset, this bias leads to a high classification error, as the digits that are tested are likely completely different from the ones the classifier was trained on.

As a conclusion of this project, we should mention the most crucial factors that play a vital role in classification procedure of handwritten digits. Regardless of the classifier or classifiers that we use, it is clear that their final performance is heavily dependent on the preprocessing of the available dataset. Removing noise and making our data as 'clean' as possible not only improves the final accuracy of the classifier but also makes the feature detection and extraction procedure much easier, avoiding any possible misbehavior due to noise. Especially in the field of image processing, like handwritten digits classification, this can lead to significantly different results. In addition, the correct representation of our data was also a very important part of our work. Despite the fact that representing our data with pixel representation did not seem to be an obvious choice at all, we finally managed to achieve the best results using it. Finally, we also have to state the importance of the training data sample in every classifier. Results, as expected, seem to be much worse in case that the training data are few in comparison to better results that are achieved for a dataset that contains at least ten times more samples. However, we still have to find the optimal training procedure.

10 Recommendations and Future Work

Two factors that are crucial for the accuracy of our system are the size of the dataset and the number of the features that we use for the development of a classifier. Minimum error (Bayes error) can be achieved for an infinite amount of data theoretically (practically impossible). In general, our learning curves show that, with the increase of the data, there is significant decrease of the error. For our case when 1000 instead of 10 samples per class are used, the error becomes approximately 3 times smaller. However, sometimes smaller but indicative training sets can lead to a better classification performance.

As far as the features are being considered, addition of extra features will not always lead to a decrease of the error. Features might be correlated and thus provide more complexity with less extra valuable information. Hence, our goal is not to find extra meaningless features, but to find features that contribute the most in the classification process. This can be depicted by the feature curves where after a certain amount of features the accuracy starts decreasing. For this reason we had to use some specific methods for feature extraction and reduction in order to reduce the dimensionality of our initial data. However there more appropriate algorithms, developed particularly for image processing, which encode interesting information into a series of numbers and act as a sort of numerical "fingerprint" that can be used to differentiate one feature from another. Such algorithms are HOG, SIFT and SURF [12, 13]. Furthermore, if we focus on class separability, Fisher mapping would also be a very good alternative for feature extraction, either instead or combined with PCA which is already used.

In addition to the above methods, also a variety of filtering techniques can be used in order to remove the noise of images. Usually, in the application domain of hand written digit recognition, salt and pepper noise is encountered due to the binary nature of the images. Another solution we didn't consider in our approach are the Median filters, which could be a solution in order to remove this noise [11].

As another approach that could be considered for improvement of the system would be the transformation of images in the z-domain or in the frequency domain (e.g. Z Transform or Fourier transform). In this way we would represent the images with the coefficients of these transformations and not using the time domain representation.

Another promising application could be a reject option. The cost of a misclassified object is likely to be higher than the cost of rejection, as a transaction with the wrong amount of money or to the wrong account holder would be more severe in comparison to a rewriting of the cheque. Hence, a reject option with the correct parameter for costs could be helpful to increase the overall performance of this process.

Finally, a trade off between time complexity and performance of a classifier exists. Long running advanced classifiers such as neural networks and SVMs are usually very computing intensive. Within our experiments they did not outperform the simple classifiers. However, a possible optimization of them (larger computational time), it would probably lead to higher accuracy.

Except for all the above, that are alternative methods, which could have been used, there are a lot of things that we would like to improve and perform further research regarding this project. We thought a lot about the preprocessing and we achieved pretty good accuracy for the pixel representation, but we did not have so accurate results for the other two representations. It would be interesting to search for better preprocessing procedures specifically for the other two representations. From the available bibliography, it is clear that feature and dissimilarities representation could have better performance in pattern recognition problems because of the structure of their space. Thus, there is much to be done in this field.

Based on the results of our experiments, we believe that a single classifier would not suffice for the classification of the hand written digits, as it depends on many factors such as the size of the training set and the number of features that are used. For sure, the choice of the classifier is a very important issue for this problem. However, it is crystal clear that the preprocessing and the feature extraction method that are used, also play a very important role for the minimization of the classification error.

References

[1] Cun Y. Le, Boser B., Denker J. S., Howard R. E., Habbard W.,Jackel L. D. and Henderson D.. Handwritten Digit Recognition with a Back-propagation Network. Advances in Neural Information Processing Systems, 396–404, San Francisco, CA, USA, 1990

[2] D. Nibaran, S. Pramanik, S. Basu, P. K. Saha, R. Sarkar, M. Kundu and M. Nasipuri, Mita. Recognition of Handwritten Bangla Basic Characters and Digits using Convex Hull based Feature Set. In International Conference on Artificial Intelligence and Pattern Recognition, 2014

[3] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. In Neural networks: The statistical mechanics perspective, 261-276, 1995

[4] C.L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: Investigation of normalization and feature extraction techniques. In Pattern Recognition Journal, 37(2).265–279, 2004.

[5] A. Khotanzad and Y.H. Hong, Invariant image recognition by Zernike moments, IEEE-PAMI, vol. 12, no. 5, 1990, 489-497.

[6] M. Sonka et al., Image processing, analysis and machine vision.

[7] Belikn M., Niyogi P. “Laplacian eigenmaps for dimensionality reduction and data representation,” Neural Computation, Vol. 15(6), pp. 1373–1396, 2003.

[8] L. Kuncheva and W. J. Faithfull. PCA Feature Extraction for Change Detection in Multidimensional Unlabelled Streaming Data. In Proceedings of the 21st International Conference on Pattern Recognition, 2013

[9] S. Theodoridis and K. Koutroumbas. Pattern Recognition, Fourth Edition. Academic Press, 4th edition, 2008.

[10] *PR tool-box for MATLAB. Retrieved 15 January 2017, from http://prtools.org/*

[11] J. Jiang and J. Shen. An effective adaptive median filter algorithm for removing salt and pepper noise in images. In Photonics and Optoelectronic Symposium, 1–4, 2010

[12] Luo Juan, Oubong Gwun. A comparison of SIFT, PCA-SIFT and SURF. pp. 143 - 152, 2009.

[13] MS Nixon, AS Aguado. Feature Extraction & Image Processing for Computer Vision, 2012.