IN4085
Pattern Recognition
Final Assignment

Authors

Dominik Harz	Grigorios Afentoulidis	Kyriakos Fragkeskos
4513649	4521862	4516923

January 31, 2016

Chapter 1

Introduction

Automatically recognizing handwritten digits is a main research field within pattern recognition. It enables the automation of processes, where normally humans are required to read the information presented and enter the data into a system. This can be utilized to recognize account numbers or the monetary amount for automatic bank cheque processing applications. There is a vast research area covering the recognition of handwritten digits and characters. A significant amount of research is conducted using the NIST dataset as well as an extended version called MNIST, [5] [6]. Research is focused on comparing different classifiers to decrease the classification error. The objective of this paper is to introduce different techniques of pattern recognition to automate the classification of individual digits. Thereby, a variety of classifiers and techniques is researched to gain insights on the classification performance and optimization possibilities. Afterwards, an approach is introduced to find the most suitable classifiers for handwritten recognition of two different scenarios. In both scenarios digits from 0 to 9 are classified, whereby each type of digit (i.e. ′0′) represents one class. The first scenario includes a training set of 200 to 1000 digits per class and the second scenario a maximum of 10 digits per class. Furthermore, different representations for the input data are used, including pixel, feature and dissimilarity representation. Overall a classification error in an independent benchmark (using $nist\_eval$) of 4.1% for scenario 1 (large training set) and 24% for scenario 2 is achieved. This is attained by the parallel combination of classifiers with product rule including feature reduction, whereby for each scenario different classifiers are combined. Moreover, the trained classifiers are applied to live test set of the author′s handwritten digits and classification errors of 61% and 21% for scenario 1 and scenario 2 respectively are attained. This article will discuss the detailed problems in recognizing handwritten digits in chapter 2. This is followed by a proposed system design to recognize these digits in chapter 3. In chapter 4 the system is implemented and experiments on different datasets are conducted. The results of the experiments are discussed in chapter 5. Furthermore, in chapter 6 the system is tested on the author′s handwritten digits, while chapter 7 represents recommendations and the conclusion of this work.

Chapter 2

Problem

The overall objective of this project is to conduct an experimental study, which shows the most suitable classifier for digit recognition for the NIST dataset. Therefore, the goal is to compare different classifiers with different representations for the input data and find the classifier that returns the minimum classification error. Also, the completion time for classifiers is taken into account. However, as this is not a primary objective of this article, experiments in relation to completion time or resource utilization of classifiers are not conducted.
The NIST dataset contains handwritten digits, which are represented as black and white images. As described in chapter 1 the dataset includes 10 distinct classes. Using this dataset, several classifiers are tested to determine the most suitable classifier for the two scenarios. For the first scenario, a large portion of the dataset is used to train the classifiers. Hence, 1000 objects for each class are received to train the classifiers. In the second scenario a small subset of maximum 10 objects per class are taken into consideration. The objective of the best classifiers is to achieve a maximum of 5% error on large datasets (scenario 1) and a maximum of 25% error on small datasets (scenario 2).

2.1 Proposed Classifiers

There is large number of classifiers available, whereby each one of the classifiers has different characteristics and limitations[8]. This section describes each classifier briefly.

k Nearest Neighbors
: k-NN is a non parametric classifier, which use the k-NN density estimation to classify a novel object. The free variable in the k-NN, is the number of the neighbours (k), which needs to be optimized using either cross-validation or other optimization techniques. The limitations of this classifiers is the density estimation is very difficult task. Also it needs a lot of data in order to find an accurate estimation of the density. Another limitation is that the discovery of the nearest neighbours is time consuming and that it has high variance and low bias[8].

Parzen Classifier
: Parzen classifier is a non-parametric equation, which use the Parzen density estimation (using kernels for smoothing) with Bayes rule, in order to make a classification. The free variable that needs to be optimized is the h (kernel size). In order to find an accurate value of h, one has to find the value of h that gives the highest log-likelihood error. The

limitations of Parzen is that the density estimation is extremely difficult and it needs lot of data in order to be estimated[8].

Logistic Classifiers.
: Logistic classifier is a linear classifier, which use linear equations to determine the decision boundaries. Logistic classifier is extremely similar with the linear discriminant classifier. The only difference is that, logistic classifier make less assumptions than linear discriminant classifier[8].

Fisher Classifier
: Fisher is a linear classifier, which aims to minimize least square error. It is simple linear classifier which works well with small datasets and it is scale insensitive[8].

Linear Discriminant Classifier
: Ldc is Bayes plug-in classifier which assumes same covariance matrix for all the classes. It is scale insensitive. The only drawback is that it needs a high amount of data in order to find the unknown parameters including mean vectors, covariance matrix and priors[8].

Nearest Mean Classifier
: Simple linear classifier which classifies each object based on the euclidean distance between the new object and the mean from each class. It assumes identical covariance matrix, is scale sensitive and very fast classifier[8].

Quadratic Discriminant Classifier
: QDC is a Bayes plug-in classifier which assumes different means and covariance matrix for each of the classes. It needs lot of data, in order to find the unknown parameters. For example, in two-class classification problem with l-dimensions, qdc has to estimate both covariance matrices, both mean vectors and both priors, i.e, $2 + (2 * l) + ((2 * l * (l + 1))/2)$ parameters[8].

Neural Networks
: Neural networks are non-linear classifiers, which were inspired by biological neural networks (human brain). Neural networks contain nodes (neurons), which are categorized in three categories, the input, the hidden and the output neurons. The input neurons just feed the data to the system, hidden neurons try to optimize the weights of the model and the output node combine the outputs of the hidden neurons. For this project, several types of neural networks like the automatic neural network classifier, random neural network classifier, train feed forward neural network by Levenberg-Marquardt rule and train feed forward neural network classifier by backpropagation are used[8].

Support Vector Machine
: Support Vector Machine is a linear classifier, which tries to find a classification boundary that maximize the margin between the support vectors. SVM is suitable for datasets with large dimensionality and small dataset. However, SVM is time consuming classifier and also when the data are not separable may lead to suboptimal results[8].

Chapter 3

System Design

For both scenarios, three representations of the input data build the basis for the classifiers. This includes pixel representation, which uses the image as it is (i.e. black and white pixels). Furthermore, feature representation based on two PRTools[1] techniques ($im\_profile$ and $im\_features$) are used to calculate image features including dimension, mean, stddev, gravity, size, center, and many others[2] [3]. In the dissimilarity representation Euclidean distance and city-block distance is utilized to create dissimilarity matrices.
Following the basic preprocessing steps are illustrated that aim to normalize the input data to normalize the classification:

1. Boxing of the image

2. Rotation of the image

3. Resizing of the image to 16x16

4. Boxing of the image

After the preprocessing, the classification for both scenarios and for all the available representations is executed, by training and testing the above classifiers without any feature reduction technique. The testing is done by using cross-validation[8]. The Parzen and k-NN classifiers are opimized by determining their variables using 10-fold cross-validation and different values of their variables. Figure 3.1 illustrates the log-likelihood of the Parzen classifier (train and test) with different values of h. The optimal value for pixel representation is $h = 0.25$ for scenario 1. The value from the test set is used, because it is more accurate than the training set. For the k-NN 10-fold cross validation for different values of k is used[8]. These optimization techniques are utilized for all the scenarios and for all the representations. Table 3.1 presents the optimal values for both h and k for Parzen and k-NN respectively in respect to all representations and scenarios.

[1]`http://PRTools.org/`
[2]`http://www.37steps.com/prhtml/prtools/im_profile.html`
[3]`http://www.37steps.com/prhtml/prtools/im_features.html`

		Pixel	Feature (profile)	Feature (features)	Dissimilarity (Euclidean)	Dissimilarity (City-Block)
Scenario 1	Parzen (h values)	0.25	0.01	0.4	0.1	0.3
	k-NN (k-values)	3	5	4	4	3
Scenario 2	Parzen (h values)	0.5	0.05	5	0.5	5
	k-NN (k-values)	3	3	2	3	3

Table 3.1: Optimized values for Parzen and k-NN for both scenarios and representations

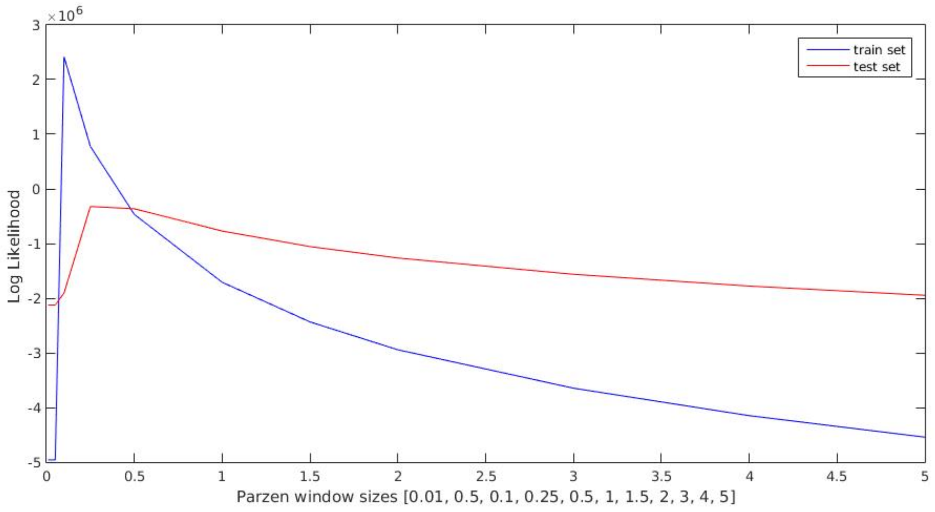

Figure 3.1: Parzen Window Size with respect to Log-Likelihood

Furthermore, feature reduction possibilities are researched. Feature extraction is applicable for both pixel and feature representation. However, feature selection is only applicable for the representation, since feature selection techniques have a significant time complexity in large dimensions[8]. Therefore it is not applicable to pixel representation, which uses $16 * 16 = 256$ features. Dissimilarity representation is not able to reduce features, because it uses dissimilarity measure between all the objects. Sequential feature selection techniques are deployed including forward, backward, floating and individual selection in feature representation. Feature extraction is realized with PCA in pixel and feature representation.
First, PCA (feature extraction) is deployed on the best three classifiers for each representation (pixel and feature) for both scenarios. As an example figure 3.2 illustrates the classification error for the best three classifiers for scenario 1 with pixel representation in relation to the appropriate number of principal components (PC). Also, figure 3.3 shows the classification error with the size of the dimensions for scenario 2 with feature representations. Those type of graphs are generated for all scenarios for the feature and pixel representation to find the new dimensions using PCA. In similar fashion as with PCA feature selection techniques are applied only for

feature representation for the best classifiers in every scenario to determine the optimal subset of features that can be used.

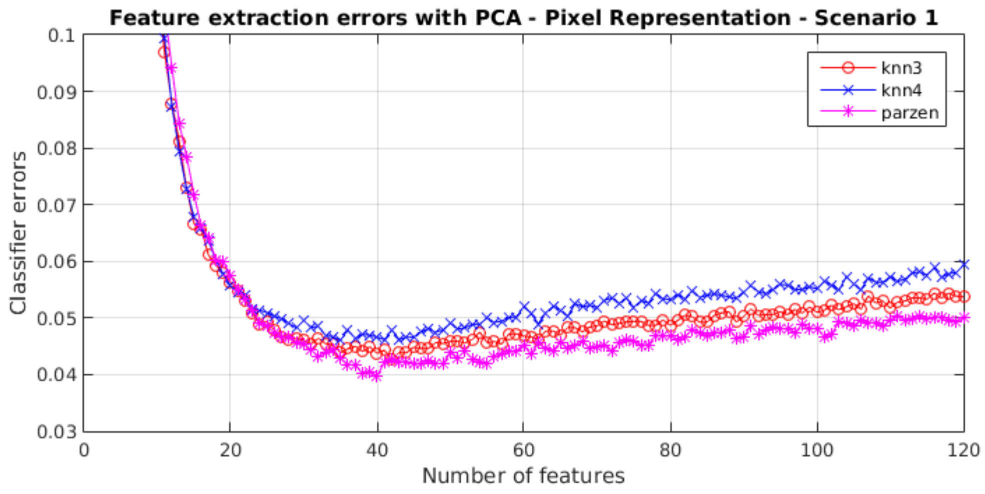

Figure 3.2: PCA results for pixel Representation Scenario 1

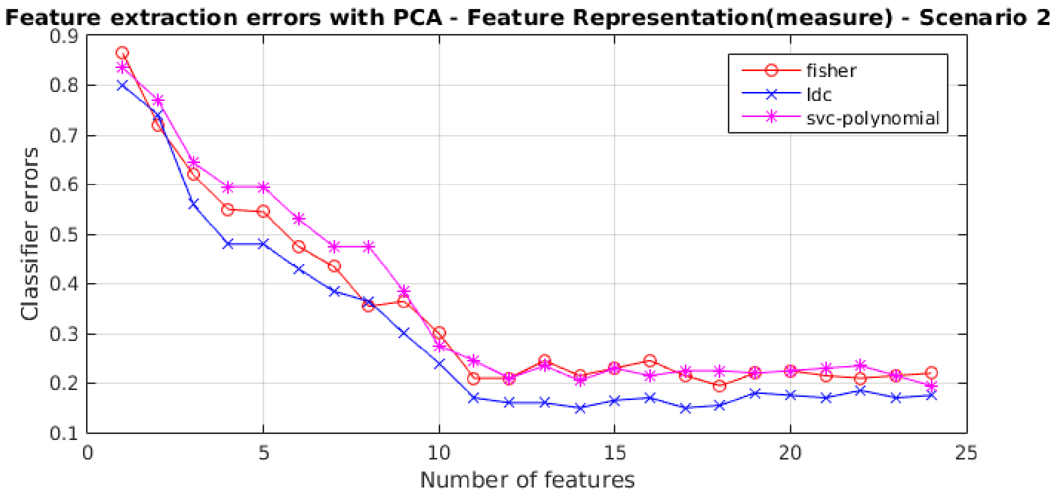

Figure 3.3: PCA results for feature Representation(im_features) Scenario 2

Moreover, classifiers are combined with different strategies and combination rules and tested. Finally, the overall best classifiers found are tested with the benchmark. The experiments and the results from the classifiers are presented in the next chapter.

Chapter 4

Experiments and Results

This chapter presents the experiments and results firstly for scenario one and afterwards for scenario two. For each scenario first the error of each classifier tested on a specific representation is presented. These classifiers have been optimized where possible as described in chapter 3. Afterwards, these experiments are repeated with feature reduction strategies and again the results are shown. As a next step, two different combination strategies are evaluated and the according errors are presented. Finally, the best classifier for each scenario is benchmarked with the $nist\_eval$ function.

4.1 Experimental Setup

The system with the different classifiers, feature reduction strategies and combination strategies are evaluated in experiments with two different datasets as described in chapter 1. The experiments are conducted within Matlab R2015b and PRTools 5.3.0[1]. The software is running on commodity hardware with Intel i5 or i7 processors and 8 GB of RAM. The implementation of different classifiers, feature reduction and combination strategies are drawn from PRTools. Every classifier or combination thereof is evaluated with 10-fold cross-validation with 10 repetitions. Cross-validation is used with these parameters to estimate the apparent error of the particular classifier and reduce overfitting[7].

4.2 Scenario 1 Evaluation

4.2.1 All representations

For scenario 1 the results of the classification error and the standard deviation for each representation are summarized in table 4.1. The three best classifiers are Parzen with kernel size $h = 0.25$, 3-NN and 4-NN for pixel representation; 5-NN, 4-NN and 3-NN for feature representation based on $im\_profiles$; logistic linear classifier, neural networks (neurc) and quadratic classifier for feature representation based on $im\_features$; Fisher, 3-NN and 4-NN for dissimilarity based on Euclidean distance; as well as 3-NN, 4-NN and 5-NN for City-block distance. Within the pixel representation comparably complex classifiers including $neurc$, $lmnc$ and svc demand high capacity of memory and their runtime is rather long on commodity hardware.

[1] `http://PRTools.org/`

As the runtime of these classifiers is above 24 hours on the available hardware, the authors decided to exclude these classifiers for this particular scenario. The same holds for svc in feature representations as well as $loglc$ in dissimilarity representation based on Euclidean distance.

		Representation				
Classifiers		**Pixel**	**Feature (im_profile)**	**Feature (features)**	**Dissimilarity (Euclidean)**	**Dissimilarity (Cityblock)**
2-nn	*Error*	0.0696	0.1428	0.3561	0.1127	0.1157
	Std	9.90E-04	8.11E-04	0.0025	0.0023	0.0024
3-NN	*Error*	0.0602	0.1258	0.3363	0.1042	0.1063
	Std	9.35E-04	0.0013	0.0022	0.0026	0.0029
4-NN	*Error*	0.0644	0.1228	0.3273	0.1034	0.1087
	Std	7.15E-04	9.14E-04	0.0014	0.0029	0.0026
5-NN	*Error*	0.0653	0.1226	0.3273	0.1088	0.1135
	Std	5.51E-04	0.0011	0.0024	0.0023	0.0024
loglc	*Error*	0.8108	0.2136	0.1286	N/A	0.3708
	Std	0.0231	9.02E-04	6.18E-04	N/A	0.0036
fisherc	*Error*	0.1904	0.2818	0.1836	0.0694	0.3718
	Std	9.25E-04	8.27E-04	7.62E-04	0.0018	0.0063
ldc	*Error*	0.1668	0.228	0.1519	0.9	0.8639
	Std	9.25E-04	5.54E-04	5.52E-04	0	0.0031
nmc	*Error*	0.2241	0.3092	0.6377	0.9	0.9
	Std	6.73E-04	7.34E-04	0.0011	0	0
qdc	*Error*	0.3253	0.1583	0.1423	0.9	0.9
	Std	0.0013	8.57E-04	0.0032	0	0
parzenc	*Error*	0.0563	0.1294	0.3134	0.9	0.9
	Std	9.15E-04	0.0016	0.0012	0	0
neurc	*Error*	N/A	0.26078	0.13983	0.74135	0.7139
	Std	N/A	7.50E-04	8.22E-04	0.01004724	0.00912
rnnc	*Error*	0.25371	0.24484	0.17063	0.22545	0.2359
	Std	0.00277	0.00304	0.00223	0.008506	0.0056
lmnc	*Error*	N/A	0.84324	0.8081	0.89035	0.8994
	Std	N/A	0.02607	0.02073	0.00639	0.0063
bpxnc	*Error*	0.21688	0.67124	0.53251	0.88665	0.8846
	Std	0.01502	0.049929	0.067845	0.012993	0.0137
SVC	*Error*	N/A	N/A	N/A	0.1276	0.2376
	Std	N/A	N/A	N/A	0.0049	0.0051

Table 4.1: Scenario 1 classification errors and standard deviation of different classifiers

4.2.2 Feature Reduction

As these classifiers are applied in high dimensionality, feature reduction with PCA and feature selection is used to optimize the classification error. For this experiments only the three best classifiers for each representations are used. The results from the previous step for PCA including the classification error and the number of principal components (#PC) are shown in table 4.2. As three classifiers for each representations are tested, the classification errors of one particular classifier is not available for each representation. This is indicated by "N/A". With PCA the classification errors are improved, except for *loglc* for *im_features* which stayed approximately the same.

		Representation		
Classifier		**Pixel**	**Feature (im_profile)**	**Feature (im_features)**
3-NN	*Error*	0.0428	0.1203	N/A
	#PC	42	15	N/A
4-NN	*Error*	0.0461	0.116	N/A
	#PC	35	14	N/A
5-NN	*Error*	N/A	0.1164	N/A
	#PC	N/A	15	N/A
parzenc	*Error*	0.0398	N/A	N/A
	#PC	40	N/A	N/A
loglc	*Error*	N/A	N/A	0.129
	#PC	N/A	N/A	22
qdc	*Error*	N/A	N/A	0.1298
	#PC	N/A	N/A	16
neurc	*Error*	N/A	N/A	0.1298
	#PC	N/A	N/A	14

Table 4.2: Scenario 1 feature extraction classification errors and principal components

Additionally, the results for feature selection including the classification error, number of features (#Features) and selection strategy are presented in table 4.3. Feature selection is only performed on lower dimensionality data sets. Feature selection results in a higher classification error for *im_features* for the tested classifier with a reduced number of features. For *im_profile* the classification errors of 3-NN and 5-NN are slightly decreased, while it is slightly increased for 4-NN.

		Representation	
Classifier		**Feature (im_profile)**	**Feature (im_features)**
3-NN (knnc)	*Error*	0.1213	N/A
	#Features	30	N/A
	Strategy	individual	N/A
4-NN (knnc)	*Error*	0.1319	N/A
	#Features	28	N/A
	Strategy	forward	N/A
5-NN (knnc)	*Error*	0.1248	N/A
	#Features	32	N/A
	Strategy	individual	N/A
log linear (loglc)	*Error*	N/A	0.1534
	#Features	N/A	21
	Strategy	N/A	individual
quadratic (qdc)	*Error*	N/A	0.2931
	#Features	N/A	15
	Strategy	N/A	individual
Neural networks (neurc)	*Error*	N/A	0.2205
	#Features	N/A	19
	Strategy	N/A	individual

Table 4.3: Scenario 1 feature selection classification errors and number of features

4.2.3 Best Classifiers

Lastly, classifier combination is applied to the best performing classifiers with the most efficient feature reduction strategy based on the previous results. The results for stacked and parallel combinations are shown in table 4.4. Sequential and trained combination demands high capacity of memory and their runtime is rather long on commodity hardware. As the runtime of these combinations is above 24 hours on the available hardware, the authors decided to exclude these combinations for this particular scenario. For stacked combination the overall best classifiers with PCA on the same representation are used, namely Parzen and 3-NN. The best classification errors are obtained from the minimum and product combiners. Furthermore, parallel combining is used with the best classifiers for each representation. However, as the best classifiers for the dissimilarity representation are running comparably long (above 24 hours) the authors decided to exclude this representation for the parallel combination. Hence, the resulting classifier is based on the PCA optimized Parzen for pixel representation, PCA optimized 5-NN for $im\_profile$ and a PCA optimized logistic linear classifier for $im\_features$. Overall the product combiner results in the lowest classification error for scenario 1.

		Combining rule			
Strategy		**Max**	**Min**	**Mean**	**Prod**
Stacked	*Error*	0.0434	0.0411	0.0433	0.0412
	Std	5.66E-04	0.00E+00	4.95E-04	7.07E-05
Parallel	*Error*	0.1133	0.0409	0.0878	0.0347
	Std	9.19E-04	2.83E-04	4.24E-04	1.41E-04

Table 4.4: Scenario 1 combination of classifiers with classification error and standard deviation

4.3 Scenario 2 Evaluation

4.3.1 All representations

Within scenario 2 the results of the classification error and the standard deviation for each representation are summarized in table 4.5. The three best classifiers are Parzen with kernel size $h = 0.5$, nmc and 3-NN for pixel representation; Parzen with kernel size $h = 0.05$, ldc and svc with exponential kernel for feature representation based on $im\_profiles$; ldc, Fisher and svc with polynomial kernel for feature representation based on $im\_features$; Fisher, ldc and Parzen with kernel size $h = 0.5$ for dissimilarity based on Euclidean distance; as well as ldc, svc with polynomial kernel and Parzen with kernel size $h = 5$ for City-block distance.

4.3.2 Feature Reduction

As these classifiers are applied in high dimensionality, feature reduction with PCA and forward, backward, floating and individual selection is used to optimize the classification error. For this experiments only the three best classifiers for each representations are used. The results for PCA including the classification error and the number of principal components (#PC) are shown in table 4.6. As three classifiers for each representations are tested, the classification errors of one particular classifier is not available for each representation. This is indicated by "N/A". With PCA the classification errors of the individual classifiers are improved. Additionally, the results for feature selection including the classification error, number of features (#Features) and selection strategy are presented in table 4.7. Feature selection is only performed on lower dimensionality data sets. Feature selection results in a higher classification error for the tested classifiers with a reduced number of features.

		Representation		
Classifier		**Pixel**	**Feature (im_profile)**	**Feature (im_features)**
3-NN	*Error*	0.265	N/A	N/A
	#PC	31	N/A	N/A
nmc	*Error*	0.265	N/A	N/A
	#PC	12	N/A	N/A
parzenc	*Error*	0.17	0.28	N/A
	#PC	13	24	N/A
fisherc	*Error*	N/A	N/A	0.195
	#PC	N/A	N/A	18
ldc	*Error*	N/A	0.26	0.15
	#PC	N/A	8	14
svc('e')	*Error*	N/A	0.265	N/A
	#PC	N/A	7	N/A
svc('p')	*Error*	N/A	N/A	0.195
	#PC	N/A	N/A	24

Table 4.6: Scenario 2 feature extraction classification errors and principal components

Classifier		Representation	
		Feature (im_profile)	**Feature (im_features)**
parzenc	*Error*	0.31	N/A
	#Features	32	N/A
	Strategy	individual	N/A
fisherc	*Error*	0.315	0.23
	#Features	15	11
	Strategy	individual	individual
ldc	*Error*	N/A	0.295
	#Features	N/A	8
	Strategy	N/A	forward
svc('e')	*Error*	0.415	N/A
	#Features	20	N/A
	Strategy	individual	N/A
svc('p')	*Error*	N/A	0.235
	#Features	N/A	19
	Strategy	N/A	backward

Table 4.7: Scenario 2 feature selection classification errors and number of features

4.3.3 Best Classifiers

Lastly, classifier combination is applied to the best performing classifiers with the most efficient feature reduction strategy based on the previous results. The results for stacked and parallel combinations are shown in table 4.8. Sequential and trained combination demands high capacity of memory and their runtime is rather long on commodity hardware. As the runtime of these combinations is comparably long on the available hardware, the authors decided to exclude these combinations also for this scenario. For stacked combination the overall best classifiers with PCA on the same representation are used, namely Parzen and nearest mean classifier. The best classification errors are obtained from the minimum and product combiners. However, the classification errors are higher than classifiers used individually. Furthermore, parallel combining is used with the best classifiers for each representation. As the best classifiers for the dissimilarity representation are running comparably long the authors decided to exclude this representation for the parallel combination. Hence, the resulting classifier is based on the PCA optimized Parzen for pixel representation, PCA optimized svc with exponential kernel for $im\_profile$ and a PCA optimized ldc for $im\_features$. Overall the product combiner results in the lowest classification error for scenario 2.

Combining rule					
Strategy		**Max**	**Min**	**Mean**	**Prod**
Stacked	*Error*	0.51	0.395	0.51	0.39
	Std	0.00E+00	7.10E-03	0.00E+00	0.00E+00
Parallel	*Error*	0.288	0.155	0.288	0.135
	Std	1.69E-02	1.51E-02	1.69E-02	1.27E-02

Table 4.8: Scenario 2 combination of classifiers with classification error and standard deviation

Classifiers		Representation				
		Pixel	**Feature (profile)**	**Feature (features)**	**Dissimilarity (Euclidean)**	**Dissimilarity (Cityblock)**
2-nn	*Error*	0.343	0.398	0.517	0.336	0.336
	Std	6.70E-03	0.0092	0.0106	0.0107	0.0178
3-NN)	*Error*	0.32	0.385	0.535	0.325	0.31
	Std	1.25E-02	0.0108	0.0118	0.0184	0.0163
4-NN	*Error*	0.322	0.422	0.52	0.338	0.348
	Std	1.48E-02	0.0162	0.0125	0.0123	0.0114
5-NN	*Error*	0.335	0.401	0.524	0.335	0.373
	Std	2.32E-02	0.0191	0.0107	0.0108	0.0283
loglc	*Error*	0.354	0.554	0.329	0.251	0.371
	Std	0.0222	0.0276	0.0208	0.0158	0.0166
fisherc	*Error*	0.369	0.465	0.215	0.244	0.375
	Std	2.33E-02	0.0242	0.0135	0.0158	0.0127
ldc	*Error*	0.9	0.368	0.172	0.244	0.258
	Std	0.00E+00	0.0187	0.0132	0.0097	0.0278
nmc	*Error*	0.279	0.39	0.642	0.336	0.37
	Std	7.40E-03	0.0156	0.0155	0.0107	0.0176
qdc	*Error*	0.9	0.607	0.399	0.9	0.9
	Std	0	0.0216	0.0185	0	0
parzenc	*Error*	0.204	0.297	0.502	0.249	0.269
	Std	7.00E-03	0.0116	0.0092	0.011	0.0129
neurc	*Error*	0.745	0.702	0.569	0.676	0.688
	Std	0.03374	0.02898	0.039	0.0334	0.03190262
rnnc	*Error*	0.624	0.486	0.35	0.42	0.426
	Std	0.053	0.02875	0.0518	0.037	0.033065
lmnc	*Error*	0.795	0.71	0.554	0.68	0.745
	Std	0.0283	0.03431	0.057	0.051	0.03064
bpxnc	*Error*	0.877	0.855	0.865	0.877	0.872
	Std	0.0149	0.029533	0.02635	0.0236	0.02201
svc	*Error*	0.297	0.503	0.278	0.34	0.276
	Std	0.0221	0.0125	0.027	0.0205	0.0201
svc('p')	*Error*	0.281	0.497	0.255	0.344	0.259
	Std	0.0145	0.0183	0.0178	0.0165	0.0208
svc('h')	*Error*	0.295	0.505	0.281	0.352	0.277
	Std	0.0172	0.0158	0.03	0.0215	0.0183
svc('e')	*Error*	0.255	0.374	0.514	0.238	0.855
	Std	0.0127	0.0143	0.0117	0.0092	0.0071
svc('r')	*Error*	0.527	0.472	0.836	0.709	0.9
	Std	0.0142	0.0187	0.0135	0.012	2.09E-16
svc('s')	*Error*	0.617	0.474	0.84	0.755	0.74
	Std	0.0221	0.0125	0.027	0.0205	0.0201

Table 4.5: Scenario 2 classification errors and standard deviation of different classifiers

4.4 Benchmark

In the previous subsection the classifiers with the minimum classification error for scenario 1 and 2 are the according parallel classifiers with product combination. These two classifiers are benchmarked with $nist\_eval$ and the results are shown in table 4.9. With an error of 4.1% for scenario 1 and an error of 24% for scenario 2 both classifiers fulfill the requirements of 5% and 25% classification error respectively. The number of objects per class used by $nist\_eval$ for scenario 1 is set to $n = 100$ and $n = 10$ for scenario 2. Furthermore, benchmarks of the previously found best classifiers with and without feature reductions are conducted. Within scenario 1, similar results are obtained with using a PCA optimized Parzen classifier. However, for scenario 2 the benchmark error for single classifiers with and without PCA is above 32%.

Scenario	**1**	**2**
Error	4.1%	24.0%
n	100	10

Table 4.9: Overall benchmark benchmark results

Chapter 5

Discussion

Finding an optimal solution for digit recognition depends on various parameters as described in the previous chapters. Firstly, the choice of representation and pre-processing results in different classification errors when applied[8]. Within the experiments the images are sized to 16X16. This balances the runtime of classifiers especially when using pixel representation. Furthermore, the size of the images also impacts the runtime of calculating the dissimilarity matrices. However, the experiments do not cover a variation of image sizes. Lecun et al. are using a 32X32 representation of images, which results in more available features [4]. Hence, a variation of pixel size can be researched in the future to evaluate a possible impact on the classification error. In regards to feature representation the possibilities of PRTools are fully used. For the dissimilarity representation other distance measurements can possibly add value. For instance the Mahalanobis distance can be utilized to measure the distance between the class distributions[8]. Especially for classes for which the mean is relatively similar but the distributions differ, this distance measurement can help differentiating the classes. This needs to be researched in further experiments.
Cross-validation in the experiments is used to separate the training and test data. In both cases 10-fold cross-validation with 10 repetitions is used. Although cross-validation results in less overfitting [8] some overfitting is still in place. With the tested classifiers the experiments show that in general simple classifiers (i.e. k-NN, Parzen) perform rather well and complex classifiers (i.e. neural networks and SVMs) require optimization. Also, only the SVM with an exponential kernel resulted in a comparably low classification error. Noticeably, in scenario 1 and 2 some classifiers completely collapse and result errors around 90% and above. When combining classifiers the experiments resulted in the lowest classification error. However, the combination strategy has a high impact on the classification error and results in considerably worse classification errors, when applied suboptimal. Others have shown that with a combination of classifiers including neural networks one can achieve very low classification errors on large datasets [4]. This results in a higher complexity of the classification procedure and is therefore also prone to possible errors within its implementation.
Feature reduction is a way than can lead to an improvement of the classification error. With PCA the experiments show that the classification error is reduced. Notably the number of principal components in pixel representation is on average reduced from 256 features to around 40 principal components in both scenarios. Additionally, for $im\_profile$ the number of principal components is around 15 resulting from 32 features. The reduction for $im\_feature$ is less and ranges from 14 to 22 principal components resulting from 24 features. On the other hand, feature selection is sensitive to the size of the dataset it is applied to. In scenario 1 feature selection

results in a slight improvement of classification error, while in scenario 2 it resulted in a decrease of the performance of the classifier. Hence, feature reduction with PCA offers a possibility to reduce the number of features, while feature selection needs to be handled carefully.
Overall, the classification error as determined with the best classifiers and cross-validation differs from the results obtained by $nist\_eval$. In scenario 1 the difference is rather small (around 0.6%) and possibly results from 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. This in nature results in a variance of results obtained. Within scenario 2 the difference between the cross-validation result and the benchmark result is comparably high (around 10%). When the combined classifier is trained on a small dataset, it results in a bias towards this particular dataset. Within cross-validation the training and test data is drawn from one dataset, assuming that this represents a dataset close to the true distribution. When $nist\_eval$ 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 to be different from the ones the classifier was trained on.

Chapter 6

Live Test

In order to better understand the performance of the classifiers that best apply to the domain of digit recognition a live test is conducted. For this test the authors have written down 10 digits for each class and then scanned it to process them. The original scanned image is shown in figure 6.1

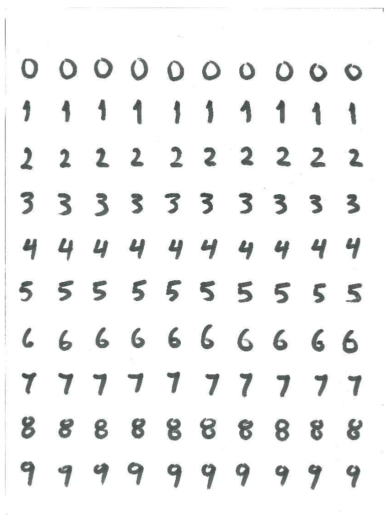

Figure 6.1: Handwritten digits for live test

In order to match the digits of the NIST data set that is used for the training phase of our classifiers a function is created which segmented the image above and binarized each digit image. Eventually, a prdatafile[1] is created out of the set of these digits. Further the prdatafile is converted into a prdataset [2] that is used as a test set for the classifiers. An example of a digit is illustrated in figure 6.2. For each scenario the best classifiers found are trained and tested with the handwritten data set to measure the classification error.

[1]`http://www.37steps.com/prhtml/@prdatafile/prdatafile.html`
[2]`http://www.37steps.com/prhtml/@prdataset/prdataset.html`

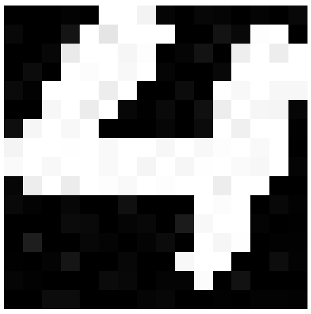

Figure 6.2: A digit example from the live test data set

6.1 Scenario 1

For scenario 1 the best performance is obtained for pixel representation and for the classifiers Parzen, 3-NN and 4-NN with the appropriate principal components based on feature extraction. For the live test first a combination of the three best classifiers of pixel representation is tested, then each classifier individually is evaluated and finally a combination of the best classifiers from every representation is used. The results are shown in table 6.1.

As it is readily seen from the results of table 6.1 there is a high error in every occasion which is always above 60%. This performance can be explained as the result of the way the digits were written. More specific a thick marker was used to write down the digits which resulted in more clumsy images as compared to those of the NIST data set. Also, errors in the process of construction of the individual digit images can be a reason of this phenomenon. The latter is the reason why the combination of the best classifiers for each representation (the best classifier performance-wise), is not the case for the live test. That means that the images have very different characteristics from those of the NIST data set. The system was trained for digit recognition in a specific field (post offices) and that may result in over-training our classifiers for this problem. Thus, one can conclude that building a generic digit recognition system which for example can also classify digits from TU Delft students is a much more challenging problem with different requirements. Especially, when one needs to decide on which dataset to train system.

Classifier	Classification Error
Parzen,knn3,knn4 (pixel representation)	0.62
knn3 (pixel representation)	0.64
knn4 (pixel representation)	0.61
parzen (pixel representation)	0.62
Parzen,knn5,loglc (all representations)	0.84

Table 6.1: Live test error for best classifiers of scenario 1

6.2 Scenario 2

Furthermore, the classifiers from scenario 2 are applied to the live data set. The approach of scenario 1 is utilized with the only difference that the best classifiers is found for feature representation with $im\_profile$ instead of the pixel representation. The results are summarized in table 6.2.

The results from table 6.2 further support the argument of overfitting described before. For this scenario the classifiers are trained with much less digits compared to scenario 1 and from the difference of classification errors the overfitting in the first case is obvious. The appearing improvement of the classification error is here justified from the fact that our classifiers are much more biased. Two more characteristics are important to notice here. First, the fact that svc utterly fails to classify the digits but ldc and fisherc come to rescue to give a much better performance when combined. That means that overall classifier combination shade classifier specific peculiarities, thus it should be preferred. Second, one notices the very good performance for the combination of the best classifiers for each representation. This low classification error is suspicious given the fact that those classifiers do not perform so well on their own, thus we concluded that this is serendipitous result which should be studied further.

Classifier	Classification Error
ldc,fisherc,svc (feature representation)	0.52
ldc (feature representation)	0.5
fisherc (feature representation)	0.52
svc (feature representation)	0.83
ldc,parzen,parzen (all representations)	0.21

Table 6.2: Live test error for best classifiers of scenario 2

Chapter 7

Recommendations and Conclusions

The size of the training data is most certainly crucial in the development of a classifier. If there is an infinite amount of data theoretically one can achieve the minimum error (Bayes error). There is a caveat though, as to the application of the pattern recognition system and the domain it will be used. Sometimes smaller but more indicative training sets can lead to better classification performance. The latter is the conclusion resulting from the tests conducted with a set of the authors' handwritten digits. In general the learning curves show that with the increase of the data there is significant decrease of the error. Furthermore, the 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 information that would help to attain good performance. Thus, the goal is not to find extra meaningless features but to find features that contribute the most in the classification process. That is better shown by the feature curves where one can see that after a certain amount of features (dimensions) the classifiers is underfitting (curse of dimensionality).

Another promising application can be the devise of a reject option. The cost of a misclassified object is likely to be higher than the cost of rejection, as a money transfer with the wrong amount of money or to the wrong account is likely cause more costs than having the customer repeat writing the cheque. Hence, a reject option can be implemented to increase the overall performance of the process. Also, a trade off between time complexity and performance of a classifier exists. The time complexity of the classifier should be low enough (based on user requirements) and high accuracy. Long running classifiers such as neural networks and SVMs are usually very computing intensive. Within our experiments they did not outperform the simple classifiers. The last observation supports the argument that within the process of building a classification system the consideration of seemingly simple classifiers is necessary. The effectiveness of the k-NN classifier for example is studied and the error bounds found are very promising [2] [1]. This further contributes to the advantages of simple classifiers that can achieve very good performance without the need of more complex techniques and methods.
However, this by no means suggest that more sophisticated solutions cannot be considered. In the domain of digit recognition from images there is a plethora of different approaches. For example in the stated problem one can also use gradient images instead of the raw digit images. The gradient images is a method for edge detection that would give a whole new feature set that might lead to better results. Moreover, denoising images is very important and many filtering techniques can achieve this. In the discussed problem salt and pepper noise is encountered because of the binary nature of the images. This can be eliminated by median filters [3]. Also the transformation of the images in the frequency domain (e.g. Fourier transform) and the

representation of the images with the coefficients of the transformation is another approach that is very promising and needs to be researched in the future.

Bibliography

[1] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[2] L. Györfi, L. Devroye, and G. Lugosi. A probabilistic theory of pattern recognition, 1996.

[3] J. Jiang and J. Shen. An effective adaptive median filter algorithm for removing salt & pepper noise in images. In *Photonics and Optoelectronic (SOPO), 2010 Symposium on*, pages 1–4. IEEE, 2010.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[5] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.

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

[7] R. B. Rao and G. Fung. On the dangers of cross-validation. an experimental evaluation. In *SDM*, pages 588–596. SIAM, 2008.

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