Pattern Recognition

Douwe Brinkhorst
Ivo van Kreveld
Vasco de Bruijn

Januari 2019

1 Introduction

This report discusses the research of a pattern recognition problem, which was done as an exercise for the pattern recognition course at TU Delft. The goal of the research was to find a method to classify handwritten digits from bank account numbers. To research this, a standard dataset put together by NIST (US National Institute of Standards Technology) was used to test the methods on.

The classification needed to be done in two scenarios, with the main difference being the amount of data available. The next sections are divided in those two scenarios, where scenario 1 is the case with 200 to 1000 samples per class and scenario 2 is the case with 10 sample per class.

Section 2 discusses the design choices that were made about the representation and the classifier used. Section 3 describes the relevant tests that were done. Section 4 discusses the conclusion of the research. Section 5 has additional recommendations that should be kept in mind while using the proposed methods.

2 Design Choices

2.1 Representation

2.1.1 Scenario 1

Since the choice of representation is based on pixels, the first feature reduction step is lowering the resolution of the image. This reduces the amount of noise in the image while preserving the geometric and spatial information in the image. It also reduces the amount of features, which makes the classifier less susceptible to the curse of dimensional and increasing the speed of the system. Subsequently, principal component analysis (PCA) is used for further dimensional reduction. PCA has the properties of maximizing the variance in the

data and making the projected data uncorrelated. Having the data uncorrelated ensures lack of information redundancies in the data while high variance means that the data is spread out better which makes it easier for the classifier to separate different classes.

2.1.2 Scenario 2

For the case where only 10 samples per class could be used, two different approaches were used.

The first approach had little preprocessing, to save computation time for more intensive methods such as AdaBoost and bagging. The image was only boxed and resized to 32 by 32 pixels.

The second approach focused on a better representation of the image. For any classifier, it is important that images of the same class look like each other. Therefore, it seems intuitive to edit images such that they have more general properties. For example, if you compare images by overlap, a '1' written slightly left from the middle and a '1' written slightly right form the middle have no overlap, since a '1' is so thin. To solve this, all numbers are moved such that their center of gravity aligns with the center of the image.

The nice property this brings is that images with numbers which are identical but in a different position will be moved such that they are in exactly the same position. Also, images that are almost identical will almost be in the same position. Besides moving to the center of gravity, rotating such that numbers would have the same rotation was also tested. Finally, to make the line thickness and the shape of lines more the same, morphological methods like dilation and erosion were tested in several combinations.

2.2 Classifier

2.2.1 Scenario 1

Even after feature reduction, the data is still quite complex and the data does not seem to be able to be described by a model. In addition, there is quite some variation in representation of the same digit. Therefore, a parametric classifier would likely have a bad performance as this requires assumptions about the distribution of the data and tries to 'summarize' the data in a classifier. Since this scenario allows for 10000 training objects and has to be trained only once, using a non-parametric classifier is a valid option, as a non-parametric classifier requires a (relatively) large training set and are slower to train.

The Nearest Neighbour classifier is known to have a good performance for a large N regardless of the distribution of the data. The KNN classifier assigns the class to an image which is taken from the images in the training set which are visually closest to the new image. Therefore, it is able to deal with the different ways to write a digit, provided that the dataset already contains an image in the same style. Because of this, we expect the KNN classifier to be the best performing classifier in this case.

2.2.2 Scenario 2

In this case, the number of objects is small in comparison to the number of features (if no feature transformation/reduction is used). This means the 'curse of dimensionality' is definitely present here.

As stated above, the digits are too diverse to describe with a parametric classifier. Even if the data was less diverse, a parametric classifier or neural network will likely struggle to generalize with this little data. A support vector machine or nearest-neighbour classifier is less likely to suffer from such generalizations as they are distribution-agnostic.

With few objects and many features it also follows that the data is likely to be (linearly) separable. Therefore, it is expected that there is little benefit to using a non-linear kernel for the SVM. Given the smaller dataset, computationally expensive methods such as AdaBoost and Bagging become more feasible.

3 Test

3.1 Scenario 1

In order to test the system, other classifiers will be used for comparison and be evaluated with cross evaluation ($N = 10000, c = 10$) in order to determine if 1-NN is truly the best classifier.

Error	1-NN	Parzen	NMC	LDC	QDC	Fisher	Log	Tree
μ_e	0.0454	0.9000	0.9000	0.1265	0.7510	0.1600	0.1932	0.4250
σ_e^2	0.0067	0.0054	0.0054	0.0081	0.0198	0.0076	0.0117	0.0177

Table 1: The error of the classifiers tested in scenario 1

The results in Table 1 show that the 1-NN classifier is indeed the best performing classifier. Using the mean error from the cross evaluation should give a reasonable estimate for the performance on novel data. Therefore, we expect that the error rate for this classifier is $\epsilon_t = 4.6\%$. When testing on the benchmark, the resulting error is $\epsilon_b = 4.4\%$. The slight improvement in error rate for the benchmark can be explained by the fact that the cross evaluation used a training set of $N = 9000$ while the benchmark used a training set of $N = 10000$.

A test was also done to determine the influence of the amount of samples on the performance. Figure 1 shows that the error rate is inversely proportional to N. Therefore having more data would increase the performance. However, the plot also shows that increasing the amount of data will have diminishing returns on the error rate.

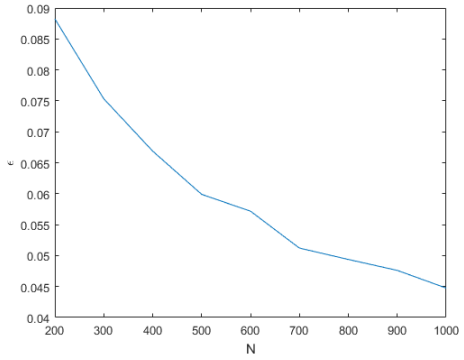

Figure 1: Plot showing the error rate ϵ against N using the test system from Scenario 1

3.2 Scenario 2

For scenario 2, both approaches described in section 2 were tested. 10 training objects per class were used, as prescribed. 100 objects per class were used in evaluation. All tests were ran 10 times with random training test data. A training time constraint (prtime) of 600 seconds was used.

For the first approach (where, as mentioned before, the only preprocessing done was boxing and resizing the image), table 2 shows the performance of the (singular) classifiers. It was predicted that the quadratic and radial basis function kernels would hardly outperform the linear kernel. In reality, these two kernels perform far worse. This may be due to overfitting. It can be seen that the neural network performs poorly, as expected.

Error	k-NN	SVM-linear	SVM-quadratic	SVM-RBF	Neural network
μ_e	0.2971	0.3369	0.4413	0.8890	0.7454
σ_e^2	0.0006	0.0027	0.0096	0.0002	0.0014

Table 2: The error of the singular classifiers tested in scenario 2

It was investigated whether the two classifiers with the best performance - k-NN and SVM with linear kernel - could be further improved by using AdaBoost or Bagging. The results of this test can be seen in table 3. The Adaboosted SVM with linear kernel is the only classifier that met the required 25% error rate. The estimated performance for novel data is expected to be 0.248. On the benchmark, it achieved an average error of 0.219 with a variance of $4.44 * 10^{-4}$.

It can also be seen that bagging performed similar or worse than the singular classifier in all cases. Bagging results in little improvement for stable algorithms, of which k-NN is one[1]. SVM is also reported to be stable, although this may not be the case for smaller datasets [2] [3].

This classifier was also combined with the more extensive preprocessing of the second approach explained in 2.1.2. This resulted in an error on the benchmark of 0.2416 with a variance of $4.82 * 10^{-4}$. This does meet the requirements, but the simple preprocessing does outperform it. This may be in part due to the larger image size it used (32 by 32 instead of 24 by 24).

Error	k-NN+AdaBoost	k-NN+Bagging	SVM+AdaBoost	SVM+Bagging
μ_e	0.3272	0.2975	0.2426	0.3812
σ_e^2	0.0005	0.0024	0.0016	0.0375

Table 3: The error of the AdaBoosted or bagged classifiers tested in scenario 2

4 Conclusion

In the first scenario with a lot of data, the tests indicate that just using PCA in combination with 1-NN gives the best test results. This verified the expectation that 1-NN performs well for a large N. Of course, it needs to be mentioned that this was only compared to the named classifiers. It could be that there are classifiers or different representations which would do better. However, as the first scenario has a lot of data, that may increase the runtime too much, so that may also not be desirable.

In the second scenario with little data, the tests indicate that using the Adaboosted linear SVM gives the best test results. Especially in this scenario though, there are a lot of possibilities of methods that would work. With such little data, it is more feasible (and more needed) to run computationally hard methods. This research covered only a small part of the possibilities. Therefore, the expectation is that there is a lot of room for improvement in this scenario.

Comparing the two scenarios also immediately shows how important enough data is. For the scenario with a lot of data, computationally hard methods may not be feasible, but they are also not needed to get an acceptable error. For the second scenario with little data, methods which are computationally harder needed to be used. This resulted in a comparable runtime, but the error of the second scenario still couldn't come anywhere near the error of the first scenario. Therefore, having a lot of data is the preferred way to train a classifier. This holds for comparing $N = 10$ and $N = 1000$. As shown in figure 1, there might be a better middle if having a low runtime is really important.

5 Recommendations

The classified digits are taken from bank account numbers and monetary values. Therefore, it is possible that not every digit has an equal chance of occurring. Especially with monetary values, one would expect that a '0' would occur more often compared to other values. By taking earlier cheque proceedings, one can

determine the frequency of each digit and use these frequencies as prior for the classifier.

The digits from the account numbers have a certain format, and are bound by validation rules. Therefore, if all digits of the bank account number are evaluated at once, the format and validation rules can be used to check for errors in the digits.

In scenario 2, the amount of training data is rather limited. This can be migrated in several ways to improve performance:

- Include the unlabeled data in the training data, and use a semi-supervised learning method to classify the data. This does have the disadvantage of using the test set as training set, which makes the error unreliable, but the performance should be better in general as more data, and therefore more information, is used in the classifier.

- Take more care with constructing the training set such that it contains a representative and high variance selection of the data.

As was discussed in the tests of scenario 1, it is also possible to decrease the number of samples to gain a better runtime, at the cost of a higher error rate. As long as the number of samples is not too low (like in scenario 2), having more training data would marginally decrease the error rate while scientifically increasing the run time. Figure 1 should be referenced to decide the optimal amount of samples to use.

One of the difficulties in recognizing digits is that there are many ways to write the same digits. One would expect in general that a person will write their digits in the same manner. Therefore, if for each digit in the training data, the author is included as well (anonymous of course) this information can be used to emphasize the digits which are from the same author. These digits should be more similar compared to random digits, which will increase the performance.

References

[1] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[2] Ioan Buciu, Constantine Kotropoulos, and Ioannis Pitas. Demonstrating the stability of support vector machines for classification. *Signal Processing*, 86:2364–2380, 2006.

[3] Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 28:1088–99, 08 2006.