Handwritten Digit Recognition — Road to Contest Victory

Norbert Jankowski and Krzysztof Grąbczewski
Department of Informatics
Nicolaus Copernicus University
Toruń, Poland
http://www.is.umk.pl/
{norbert|kgrabcze}@is.umk.pl

***Abstract*— With growing amount of data gathered nowadays, the need for efficient data mining methodologies is getting more and more common. There is a large number of different classification algorithms, but choosing the best one for given data is still a difficult task. Thanks to different data mining contests we can gather lots of meta level information about classification problems and strategies leading to optimal (or close to optimal) solutions. One of the contests was organized in parallel with the ICAISC'06 conference held in Zakopane. We took part in it, and our model classified the test data with the highest accuracy. The process which led to the winner model was not simple – it required multiaspect analysis of the data and different algorithms (from the point of view of suitability to the data). This article presents our road to the winner model with numerous comments on both successful and unsuccessful efforts. It also presents our model testing methodology, which always plays important role in the pursuit of accurate and well generalizing models.**

I. Introduction

Thorough analysis of particular data always requires using a multitude of different techniques. The data set we analyze here defines a classification task, so we need to:

- apply **different classification** algorithms,
- try **different data transformations** before the classification stage (from some simple and basic ones to some advanced functions constructed for the particular task),
- apply a **reasonable testing methodology** (for model validation).

To efficiently search for accurate models we need a data mining environment facilitating complex models construction, easy application of many learning algorithms of different types, performing validation in a simple way and possibly providing some tools for meta-level learning (like searching in the space of models).

Our efforts were supported by the GhostMiner system, which is a general tool for data mining developed by our team in cooperation with FQS Poland [1]. The system provides all the functions mentioned above. In the area of meta-learning it contains a simple parameter search algorithm, which although performing a greedy search is very helpful in exploring the space of model parameters.

II. The contest data

The data we analyze here was prepared by organizers of the ICAISC'06 conference[1]. It was the subject of a handwritten digit recognition competition organized in parallel to the conference. The organizers extended the data originally prepared by members of the Bogazici University, Istanbul, Turkey [2], [3]. The resulting set of data vectors was split into two parts. One of them contained 80% of the data (5036 vectors) and played the role of the training set. The rest (20% – 1263 examples) of the data became available at the time of the contest adjudication and served as the test set to estimate the accuracy of models on (so called) unseen data[2].

Originally the data vectors were prepared in two versions: dynamic and static. The dynamic representation of a digit consisted of a number of 2D coordinates corresponding to pen movement. This form of the data has not been available to us. The static representation has the form of a bitmap resulting from blurring 32 by 32 pixel monochromatic images and then reduction of resolution to 8 by 8 pixels. Each final pixel is described by an integer from the interval $[0,16]$, since the value is the number of black pixels in an appropriate 4 by 4 pixel part of the blurred image. The data preprocessing stage is depicted schematically in figure 1, which we copied from [2] with authors permission. The original split of the data set was different than the contest one. The authors divided the data into four sets: one for training (1934 examples), one for validation (called cross-validation in [2] but such name is misleading; this set consists of 946 examples) and two sets for testing (one called *writer-dependent* with 943 examples and the other *writer-independent* with 1797 data vectors—the former one contained digits written by the same 30 authors, who wrote the digits of the training and validation sets, and the latter contained digits written by another 14 authors).

Some examples of data vectors are shown graphically in figure 2. For each class (there are 10 classes representing digits $0, 1, \ldots, 9$) we show two examples: one very readable and one of the less "obvious" cases). The figure shows 8 by 8 pixel

[1] ICAISC'06 was the Eighth International Conference on Artificial Intelligence and Soft Computing, held in Zakopane, Poland in June 2006.

[2] During the contest, the test part was provided without class labels—the labels were disclosed after the adjudication.

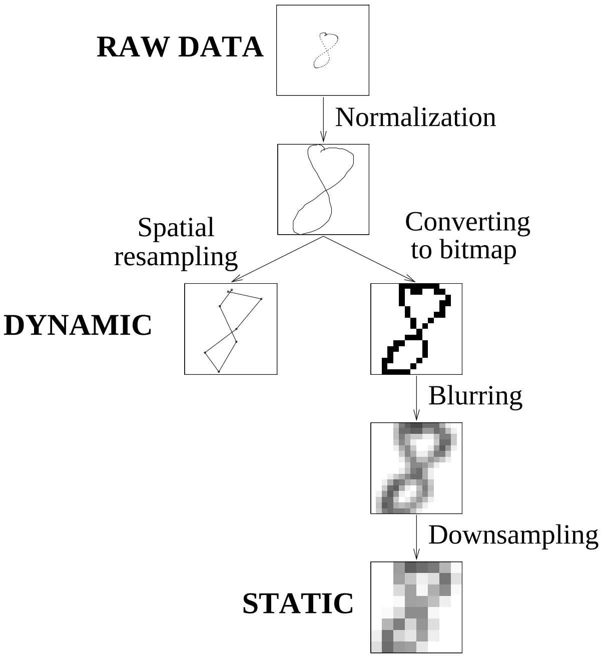

Fig. 1. Data preprocessing schema – the figure comes from [2] (included with permission of the authors).

maps representing the 64 feature values with different degrees of grey—white squares present the value of 0 and black ones the maximum value of 16. As it can be seen in the figure, the classification decisions are not always clear for human, so we should not expect the computational intelligence (CI) classifiers to be 100% correct.

III. Testing methodology

Prediction of which model will be the most accurate when applied to data unavailable (unseen) during the analysis is not a simple task. We must validate models we create, but the exact way of doing it is not self-evident. One of the most popular is to detach a validation set and estimate the accuracy on unseen data by the accuracy obtained for this validation set. This is a simple method, which does not consume much computation time, however it is not a reliable validation. To see this consider a classification task and a family of models, which obtain similar accuracy on the training data, but have different decision borders—for example we can think of simple linear discriminant models with different decision borders. Such models will usually obtain different results for the validation data set. If such family is large, the probability of finding a model very accurate in classification of the validation data is quite high, so trying large number of candidate models leads to small validation errors, but not necessarily guarantees small errors on unseen data in general. As an extreme example, we can think of a family of models generated randomly. If we tried many times, we would certainly get an accurate model for validation data, but such technique is no longer a validation, but rather learning on the validation data.

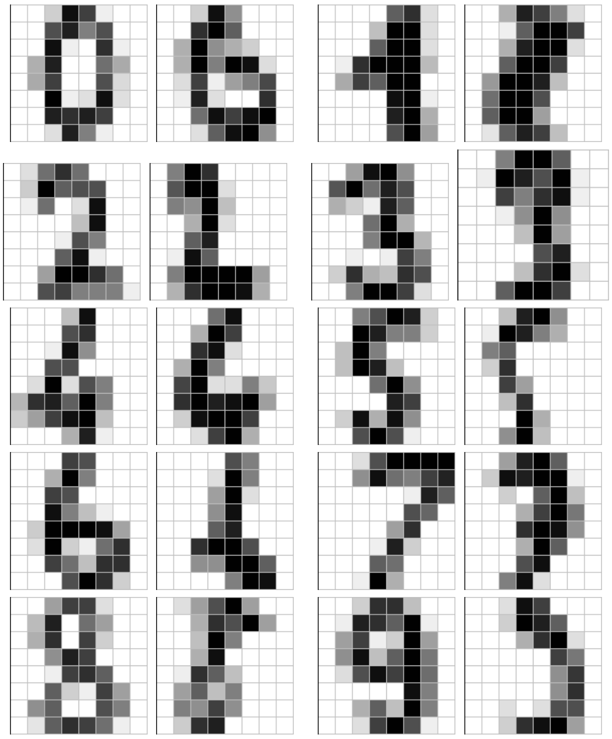

Fig. 2. Training data examples.

The same comments apply to classification results published in numerous articles, where an accuracy obtained for a separate test set is presented, e.g. many publications concerning some of the UCI repository data [4]. New models improve classification of the test data by one or two examples, and are commented as significantly better models while there is no statistical evidence to claim so. It must be emphasized, that **in such cases the test set is in fact an extension of the training set**—although it is not used directly in the adaptive processes of model training, it is used for final model selection which must be seen as a special kind of training. As a consequence, such results overestimate the accuracy on unseen data.

A separate test set is a very good solution in the case of different competitions, but it is a completely different situation than in the case of validation. The competitors can not try multiple models and select those which obtain the best results for the test set, because the contest organizers do not publish test data labels before the adjudication.

Much more reasonable way of model validation is based on the cross-validation (CV) technique. A CV performed within the training set provides an estimation of the accuracy for unseen data (the average accuracy obtained for the validation parts of CV folds). Although such CV estimates are not perfect (the limited number of data affects the independence of the CV results), they are much better than those measured for a single validation set, because the validation part in each fold is different and determined randomly, so it is much more probable that a method obtaining high CV scores provides good generalization. Moreover, in most interesting cases, we can repeat CV test to overcome the possible problem of accidental distribution.

In our approach, we used different numbers of folds in CV. The larger the number of CV folds, the more computation time is required, but the more similar training conditions are provided. On the other hand, the larger number of CV folds the less independent are the models generated in the CV process, because the common part of the training sets gets larger. Thus, determination of the optimum number of folds is not trivial.

Thanks to cross-validation we obtain the information about average test accuracy and also about the variance of these results. High average test accuracy is not the only aim we strive to. To be confident about high accuracy on the contest test set, we need a method showing small variance within the CV. Hence, our model selection criterion is usually not just the value of average accuracy, but the value decreased by a quantity related to the standard deviation:

$$S_M = A_M - \alpha\sigma_M, \tag{1}$$

where A_M is the average accuracy obtained for method M, σ_M is the standard deviation and α is a control parameter (in our approaches we usually set $\alpha = 1$). Such technique is sound with statistical approaches like hypotheses testing.

IV. Toward the winner solution

The starting point of the search for the best model is to test different base methods. By the base methods we mean a group of complementary methods which are derived from different computational intelligence fields such as machine learning, statistics or neural networks. The complementarity constraint is very important because otherwise, the search for optimal (or suboptimal) solution may take unnecessarily much time or may finish with poor results.

During the search procedure, we have always paid the most attention to the criterion (1) to select the most promising (i.e. most accurate and stable) methods.

In the first phase of searching the following methods were tried: k Nearest Neightbors [5], Naive Bayes [6], Support Vector Machine [7], [8], [9], SSV decision tree [10], [11], NRBF [12] and FSM [13].

A. Base methods

The results presented below are the validation results of 10-fold cross-validation. In most cases the CV procedure was repeated 10 times and results (errors and standard deviations) were averaged. Such tests are denoted by **XCV**. In the case of just a single CV test the label **CV** is used. The label **TE** stands before the error obtained on the testing part of data which was calculated after the contest, because the class labels were not available before.

The first tested method was the kNN. Typically the k is set to 5 at the first trial. The results are

kNN, $k = 5$	
XCV: 0.031 ± 0.007	TE: 0.033

In some sense this result can be used as a reference.

In the next step we tried to determine the optimum value of k via internal cross-validation. Internal cross-validation means that it is run within the training data. It turned out, that the range of the optimum k is more or less the interval $[3,6]$. So, the initial value was quite a good try.

The kNN was also tested with different distance measures like Minkovsky (with different scales, including 1 for Manhattan), Chebychev and Canberra. However the results were similar or worse than for standard Euclidean metric.

In another test the training data set was standardized before running the kNN. The standardization was performed separately for each feature (*per feature standardization*). The results are

Std $\mid$ kNN, $k = 5$	
XCV: 0.045 ± 0.009	TE: 0.044

It means that the standardization made the accuracy deteriorate. This is not surprising, because each feature represents *intensity* of a given region of digit. For example the top left corner will show significantly less variance than some pixels in the middle of the image, and this is a precious information we lose with per feature standardization. Other methods like *per data set standardization* (each feature is scaled with regard to the overall average and variance within the data set) or no scaling at all are much more useful.

Our second base method was Naive Bayes (NB). Unfortunately, it is very rare when the NB is among the best methods. The results without and with standardization are:

Naive Bayes	
XCV: 0.848 ± 0.012	TE: 0.853

Std $\mid$ Naive Bayes	
XCV: 0.161 ± 0.018	TE: 0.154

The third method, we tried, was Support Vector Machine (SVM). Our implementation of SVM is based on the SMO algorithm proposed by Platt [14] with modifications proposed by Keerthi [15]. SVM is a binary classifier and to use the SVM for 10-class problem a committee of SVMs must be used. For most benchmarks it is not important whether to use the *one-class against the rest* scheme (building N classifiers, where N is the number of classes) or to use *one class against one class* and to build $\binom{N}{2}$ classifiers. The results for the *one against the rest* technique and Gaussian kernel are as follows:

SVM, G	
CV: 1.0 ± 0.00	TE: -

Std $\mid$ SVM, G	
CV: 0.217 ± 0.018	TE: -

So poor results have two reasons: the first is the inadequacy of Gaussian kernel parameters (C was set to 1 and Gaussian dispersion to 0.1) and the second is that standardization is not a good idea in this case (as pointed out above). The results of searching for the optimal Gaussian parameters are presented in the following section.

In the case of linear kernel for SVM the results were better but still not so interesting:

SVM, L	
XCV: 0.12 ± 0.013	TE: -

Another tested method was the normalized RBF (NRBF). In this case we observe similar behavior as in the case of SVM—bad influence of inadequate Gaussian basis function parameters and the effect of superfluous standardization:

NRBF	
CV: 0.90 ± 0.0008	TE: -

Std | NRBF	
CV: 0.095 ± 0.012	TE: 0.086

Separability of Split Value (SSV) is a criterion used mainly for decision tree construction. Although the algorithm is independent of standardization and often generates compact and accurate trees, it is not succesful in this case:

SSV	
XCV: 0.14 ± 0.018	TE: 0.14

We have also applied FSM neural network algorithm to the problem. The adaptive process of FSM adjusts its architecture to the complexity of the problem. The results are better than all other presented so far, except those of the kNN model:

FSM	
XCV: 0.043 ± 0.043	TE: 0.042

In most cases, the efforts to improve the results presented above were fruitless, however for some methods a meta-search for appropriate parameters was successful.

B. Pursuit of optimal parameters

Finding optimal parameters of SVM can be tricky. We ran several meta-search processes to check a broad range of values. Figure 3 presents 5-fold CV test results for different values of the Gaussian kernel dispersion. The two lines with points represent the predicted accuracy and accuracy minus standard deviation (according to the criterion 1) respectively. The Gaussian dispersion in SVM behaves exponentially, so we tried the range $[-13,0]$ of powers of 2. It can be seen that the highest results are obtained when the value of bias is around 2^{-10}. Now it is clear why the initial dispersion of Gaussian kernel ($0.1 \approx 2^{-3.32}$) was so bad.

Similarly the values of the C parameter of SVM can be examined. This parameter also shows exponential behavior and again a range of powers of 2 was checked. In this case the range was $[0,10]$. Figure 4 presents the dependence between the (logarithm of) C and validation accuracy (and stability).

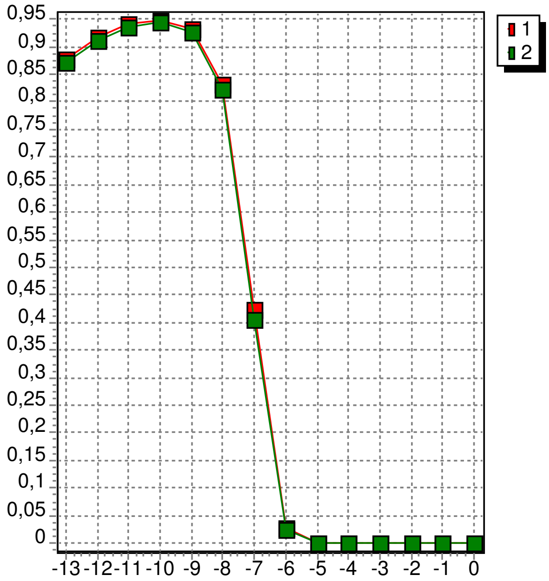

Fig. 3. Meta-parameter search for the bias of SVM's Gaussian kernel function.

The optimum values of C are in the area of 2^4 (though larger values should also work successfully), so the initial value of 1 was also far from optimal.

The SVM with new values of its free parameters yields much better results:

SVM, 16, 0.001	
XCV: 0.0377 ± 0.0075	TE: 0.028

All the interesting results we have obtained with SVM concerned the Gaussian kernel, so here and in further descriptions we write just SVM (instead of SVM, G).

We have also run a meta-search for the dispersion of normalized Gaussian basis function of NRBF. The results are presented in figure 5. As in the previous cases a range of powers of 2 was investigated. Finally the value $0.0115 \approx 2^{-6.44}$ was chosen. The optimization decreased the error to 2.84%:

NRBF	
XCV: 0.0284 ± 0.0069	TE: 0.0285

C. Selection and extraction of features and prototype vectors

We have tested a number of feature selection methods (feature selection based on correlation coefficient, F-score or SSV criterion [16]) but they were not useful because of the information loss (each feature represents a pixel and only the corner-pixels are less important but still not useless). Principal Components Analysis (PCA) was not useful either. After PCA the results were not worse, but to obtain similar results as without PCA nearly all PC's must be used, as it can be seen in figure 6.

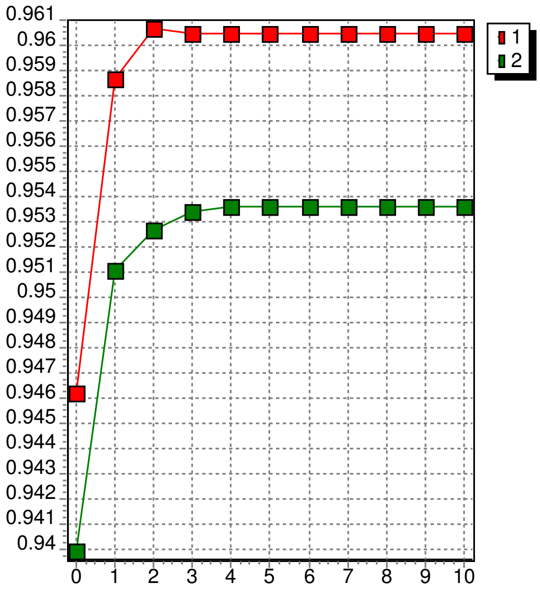

Fig. 4. Meta-parameter search for the C of SVM method.

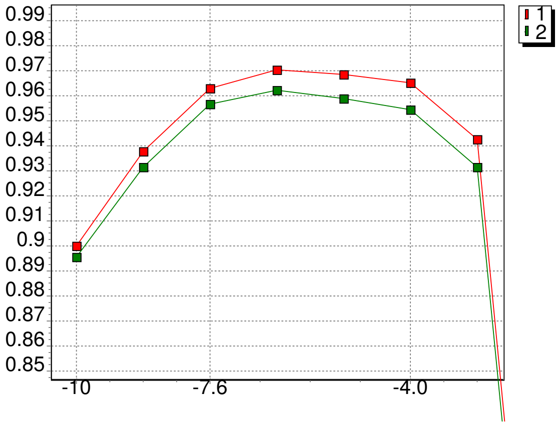

Fig. 5. Meta-parameter search for the bias of NRBF's Gaussian basis function.

Some prototype selection algorithms were also investigated. We have tested the Explore [17], DROP [18], ENN [19] and LVQ with different numbers of neurons [20]. The error with Explore was over 10%. The DROP3 was significantly better but still not satisfactory:

Drop3	
CV: 0.056 ± 0.0059	TE: 0.058

The results of ENN are slightly below those of kNN:

ENN	
CV: 0.035 ± 0.005	TE: 0.0387

The prediction of performance of LVQ with different numbers of neurons can be seen in figure 7. The conclusion is that, in this data set, there are no simple sets of prototypes, which would offer high classification accuracy. The test results for 3000 and 5000 of neurons are the following:

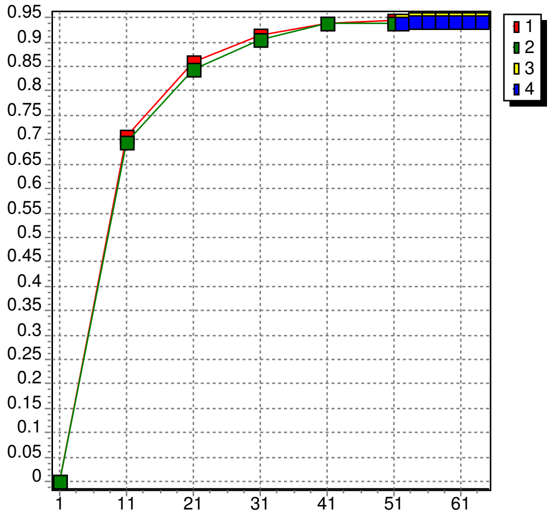

Fig. 6. SVM performance on subsequent collections of PCs.

LVQ 3000	
XCV: 0.034 ± 0.0083	TE: 0.026

LVQ 5000	
XCV: 0.0323 ± 0.0076	TE: 0.029

D. Committees

Successful models combined into committees may improve and stabilize their results. We have tested several types of committees with different configurations of models.

One of the simplest kinds of committee is based on the idea of voting (each committee member has a single vote and all the votes are equally important). A bit more advanced rule defines a *weighted committee*, where in the place of voting scheme we calculate the probabilities of belonging to the classes on the basis of the probabilities obtained from submodels (F_j):

$$p^w(i|\mathbf{x}) = \frac{1}{N}\sum_{j=1}^{N} p(i|\mathbf{x},F_j). \qquad (2)$$

Here $\mathbf{x}$ is an observed vector, $p(i|\mathbf{x},F_j)$ is the probability that vector $\mathbf{x}$ belongs to i-th class according to submodel F_j, and $p^w(i|\mathbf{x})$ is the probability that $\mathbf{x}$ belongs to i-th class according to the committee.

A weighted committee of three different prototype selection schemes: Explorer, DROP3 and LVQ (with 1000 neurons), gave quite interesting though not the best results:

CommW[Explorer,DROP3,LVQ1000]	
CV: 0.036 ± 0.0073	TE: 0.037

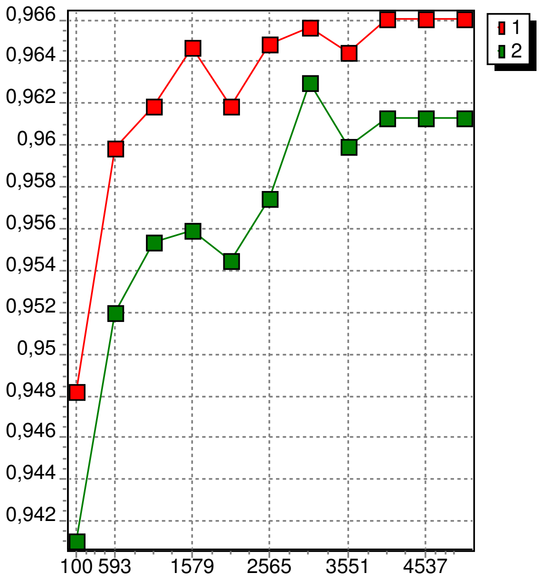

Fig. 7. Meta-parameter search for the number of neurons in LVQ method.

Another two weighted committees, worth a mention, combine SVM with NRBF and SVM with kNN respectively:

CommW [SVM+NRBF]
XCV: 0.0216 ± 0.0067 TE: 0.0174

CommW [SVM+kNN]
XCV: 0.0215 ± 0.0057 TE: 0.0206

The weighted committee composed of kNN and NRBF is significantly worse than the previous two:

CommW [kNN+NRBF]
XCV: 0.027 ± 0.008 TE: 0.028

As it was mentioned above the SVM for multi-class problems may be used in a voting scheme with SVMs trained on pairs of classes. Each of the $\binom{N}{2}$ SVM models has one vote for its winner class. The class which collects the highest number of votes becomes the final winner of the committee voting. This scheme ($\binom{N}{2}$ SVM) turned out to be the first our model to break the error threshold of 2%:

$\binom{N}{2}$ SVM
XCV: 0.0177 ± 0.005 TE: 0.0119

To stabilize and sometimes even obtain better performance a CV committee can be used. CV-committees consist of submodels trained the same way as in the CV test, but the models built, compose a committee classifier. The CV-committees were successfully used by us in [12], [21]. To escape impasses during classification 11-fold CV was used in place of the typical 10-fold CV. CV committee of $\binom{N}{2}$ SVM produced just a bit better result:

CVC [$\binom{N}{2}$ SVM]
XCV: 0.0175 ± 0.0056 TE: 0.0111

The weighted committee of $\binom{N}{2}$ SVM with NRBF were not so successful as similar committee with one class against the rest scheme:

CommW[$\binom{N}{2}$ SVM + NRBF]
CV: 0.0261 ± 0.0071 TE: 0.027

A number of committees with local competence (as proposed in [21]) were also tried, but without significant improvements. For example the above committee in its local competence version (where submodels are turned into CV committees):

CommCompW[CVC[NRBF]+CVC[$\binom{N}{2}$ SVM]]
CV: 0.0262 ± 0.0069 TE: -

Some feature extraction ideas were tested, but most without significant results. These include the *island-score* (the number of white regions in a given digit) and vertical and horizontal densities.

E. Darkening

It is easier to come up with new ideas, when we know as much as possible about the results obtained so far. Although we found quite accurate models with some methods applied to raw data, we started looking for some data transformations to get even better results. The analysis of erroneous test cases within cross-validation showed that some data vectors are easy to classify visually, but still they are erroneous cases. It was a consequence of the fact, that these vectors coordinates were smaller than for most other vectors. In other words the digits were brighter than others—sometimes the largest value in a vector was 9, while the norm is that the most intensive pixels have the value of 16. We guess that the reason behind the differences is the method of blurring used by the contest organizers, which was probably different than the one applied by the original authors. As a result digits with the same contours but with normal darkness of the pixels are quite far from their brighter copies in the sense of most distance measures. This brought the idea that it is worth to normalize the brightness of the pixels within each vector. It is a special kind of normalization since it concerns vectors instead of features. We decided to use a linear transformation with a threshold, which prevents from values greater than 16:

$$f_k(x_i) = 16 \cdot \max(1, \frac{x_i}{v_k(\mathbf{x})}), \qquad (3)$$

where x_i is the ith coordinate of vector $\mathbf{x}$ and $v_k(\mathbf{x})$ is the minimum of 1 and the kth minimum coordinate of $\mathbf{x}$. k is the parameter controlling the transformation, which must be within the range from 1 to the dimensionality of $\mathbf{x}$—the smaller k, the stronger darkening of the image (very small k should force v_k to be 1, and the pixels will get either 16 or 0 values). The transformation may be called *darkening*, because it reduces the brightness of some digit images.

To find the optimum number of k for the darkening transformation followed by a classification algorithm we need to perform some meta-search. For different classifiers we obtained $k = 56$ and $k = 59$ as the optimum values.

Two examples of exceptionally bright digits are presented in figure 8. They were selected to show how the 3NN classifier improves its results thanks to the brightness normalization. Bright digits usually seem more similar to other bright digits, than to darker digits representing the same class. The figure shows bright and darkened (with $k = 56$) versions of two examples (vectors 2576 and 4694), and their closest neighbors. The top two rows correspond to the first example, the remaining two to the second. In the leftmost column we see the illustrations of bright and dark versions of the examples. The middle column shows the nearest neighbors of the digits and the right-hand column the second nearest neighbors. The neighbors are determined on the basis of Euclidean distance to the analyzed example. It can be seen, that in the case of the first example, the bright 5 is closer to a bright 3 (example 1986, distance equal to 18.7) and to a bright 0 (example 134, distance: 21.0) than to other examples of digit 5. After darkening, the case 2576 moves in the feature space toward other examples of 5, so that the nearest two neighbors also belong to class 5 (examples 2978 and 2986 with distances of 27.1 and 32.0). Notice that the distances to another bright digits are significantly smaller than the distance to the nearest neighbor after darkening. The second example is also rectified by darkening, however it is not so evident as in the case of the former example. It is a bright 9 whose two nearest neighbors are bright representatives of class 3 (examples 1708 and 1958 with respective distances of 17.7 and 21.4). After darkening the nearest neighbor is another instance of 9 (4603 with Euclidean distance 26.9). The second nearest neighbor of the darkened 9 represents class 3 (case 1761 with distance 27.3).

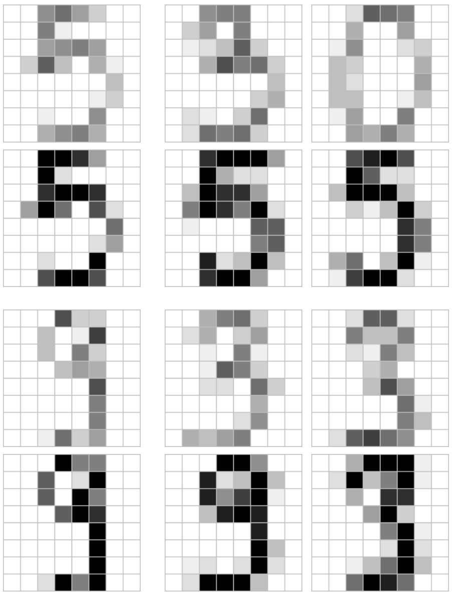

Fig. 8. Brightness correction examples.

The brightness normalization improves test accuracies by 0.1%-0.5% for most of the models we have tested. The differences (although not very large) are confirmed with the paired t-test to be statistically significant.

The first example of positive influence of darkening is the transformation performed with $k = 56$ combined with the $\binom{N}{2}$ scheme of SVMs:

Dark 56 $\mid$ $\binom{N}{2}$ SVM	
XCV: 0.0163 ± 0.0063	TE: 0.0103

As presented above, without darkening the CV error was 0.0177.

Another example of the influence of darkening is its combination with the weighted committee of NRBF and SVM:

Dark 56 $\mid$ CommW [NRBF + SVM]	
XCV: 0.0194 ± 0.006	TE: 0.0166

In this case darkening also decreased the error while preserving the standard deviation.

Another promising value of k for darkening was 59 which also offered a decrease of the error of $\binom{N}{2}$ with SVM:

Dark 59 $\mid$ $\binom{N}{2}$ SVM	
XCV: 0.0159 ± 0.0055	TE: 0.0103

To obtain higher stability a CV committee was built on the basis of $\binom{N}{2}$ SVM. Again the error got a bit smaller:

Dark 59 $\mid$ CVC[$\binom{N}{2}$ SVM]	
XCV: 0.0151 ± 0.0057	TE: 0.0103

Finally, a weighted committee of two models: a $\binom{N}{2}$ SVM and a CV committee of $\binom{N}{2}$ SVM was composed.

Dark 59 $\mid$ CommW[$\binom{N}{2}$ SVM + CVC[$\binom{N}{2}$ SVM]]	
XCV: 0.01499 ± 0.006	TE: 0.0095

Although we should not expect a significance in the differences between (at least) the last three models presented, the last one is the winner of the competition. It misclassifies just 12 test instances. The kNN with $k = 5$ misclassifies 42 instances. So the best model outperforms the kNN over three times in an absolutely fair test.

V. Conclusions and future plans

The way to the winner model was not straight or easy. The final solution was a consequence of many different types of experiments. It can not be expected that for a real world, nontrivial data, a single model will solve the problem with satisfactory results.

There are so many different adaptive methods and new ones are still emerging. Now the most important problem is to be able to find the methods (and their parameters) which provide the best models of given data. Thus, the procedures of model searching will get more and more important. Nowadays meta-search is usually performed by a human with some help of computational intelligence tools, but already now we feel a strong need for automatization of such processes.

In meta-learning it is very important to observe carefully the results of tests at each step of the search process. We need to learn how these results point the most promising directions of further steps. We need to learn more about the ways we search for attractive solutions and try to convert the knowledge into formal procedures. Our experience augmented with the possibility of performing tests thoroughly and systematically should bring very successful meta-learning techniques.

We have used some elements of meta-learning in our model searching procedures. We still work on more abstract, much more exhaustive and smart meta-learning which can be applied to different tasks, and hopefully will soon compete with humans.

Acknowledgement

The research is supported by the Polish Ministry of Science with a grant for years 2005–2007.

References

[1] N. Jankowski, K. Grąbczewski, and W. Duch, *GhostMiner 3.0*, FQS Poland, Fujitsu, Kraków, Poland, 2004.

[2] F. Alimoglu and E. Alpaydin, “Combining multiple representations and classifiers for pen-based handwritten digit recognition,” in *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR 97)*, Ulm, Germany, August 1997.

[3] E. Alpaydin and C. Kaynak, “Cascading classifiers,” *Kybernetika*, vol. 34, no. 4, pp. 369–374, 1998.

[4] C. J. Merz and P. M. Murphy, “UCI repository of machine learning databases,” 1998, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[5] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[6] T. Mitchell, *Machine learning*. McGraw Hill, 1997.

[7] V. Vapnik, *Statistical Learning Theory*. New York, NY: Wiley, 1998.

[8] B. E. Boser, I. M. Guyon, and V. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, D. Haussler, Ed. Pittsburgh, PA: ACM Press, 1992.

[9] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[10] K. Grąbczewski and W. Duch, “A general purpose separability criterion for classification systems,” in *Proceedings of the 4th Conference on Neural Networks and Their Applications*, Zakopane, Poland, June 1999, pp. 203–208.

[11] ——, “The Separability of Split Value criterion,” in *Proceedings of the 5th Conference on Neural Networks and Their Applications*, Zakopane, Poland, June 2000, pp. 201–208.

[12] K. Grąbczewski and N. Jankowski, “Mining for complex models comprising feature selection and classification,” in *Feature extraction, foundations and applications*, I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Eds. Springer, 2006, pp. 473–489.

[13] R. Adamczak, W. Duch, and N. Jankowski, “New developments in the feature space mapping model,” in *Third Conference on Neural Networks and Their Applications*. Kule, Poland: Polish Neural Networks Society, Oct. 1997, pp. 65–70.

[14] J. C. Platt, “Fast training of support vector machines using sequential minimal optimization,” in *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA.: MIT Press, 1998.

[15] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, “Improvements to Platt’s SMO algorithm for SVM classifier design,” *Neural Computation*, vol. 13, pp. 637–649, 2001.

[16] K. Grąbczewski and N. Jankowski, “Feature selection with decision tree criterion,” in *Fifth International conference on Hybrid Intelligent Systems*, N. Nedjah, L. Mourelle, M. Vellasco, A. Abraham, and M. Köppen, Eds. Brasil, Rio de Janeiro: IEEE, Computer Society, Nov. 2005, pp. 212–217.

[17] R. M. Cameron-Jones, “Instance selection by encoding length heuristic with random mutation hill climbing,” in *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, 1995, pp. 99–106.

[18] D. R. Wilson and T. R. Martinez, “Reduction techniques for instance-based learning algorithms,” *Machine Learning*, vol. 38, pp. 257–286, 2000.

[19] D. Wilson, “Asymptotic properties of nearest neighbor rules using edited data,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.

[20] T. Kohonen, “Learning vector quantization for pattern recognition,” Helsinki University of Technology, Espoo, Finland, Tech. Rep. TKK-F-A601, 1986.

[21] N. Jankowski and K. Grąbczewski, “Heterogenous committees with competence analysis,” in *Fifth International conference on Hybrid Intelligent Systems*, N. Nedjah, L. Mourelle, M. Vellasco, A. Abraham, and M. Köppen, Eds. Brasil, Rio de Janeiro: IEEE, Computer Society, Nov. 2005, pp. 417–422.