A Statistical Review of the MNIST Benchmark Data Problem

Jiří Grim and Petr Somol [*]
Institute of Information Theory and Automation
of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

The recognition of MNIST numerals is discussed as a benchmark problem. Applying the probabilistic neural networks to MNIST data we have found that the training and test set have slightly different statistical properties with negative consequences for classifier performance. We assume that the frequently used extension of MNIST training data by distorted patterns improves the recognition accuracy by creating images similar to the atypical test set numerals. In this way the benchmark experiments may be influenced by the external knowledge about the hand-written digits and the comparative value of the benchmark becomes more or less limited to recognition of MNIST numerals. As a more generally applicable benchmark model we propose recognition of artificial binary patterns generated on a chessboard by random moves of the pieces rook and knight.

Key Words: MNIST benchmark, multivariate Bernoulli mixtures, EM algorithm.

[*]E-mail address: grim(somol)@utia.cas.cz

1. Introduction

The MNIST collection of handwritten digits is one of the most frequently used benchmark databases to test and compare different pattern recognition methods and machine learning algorithms. The website [9] reports 69 references with a brief characteristic of the underlying recognition technique, the achieved accuracy and a link to the corresponding paper.

The MNIST database was constructed from the original NIST Special Databases SD3 and SD1 which contain unformatted binary images of handwritten digits. As the SD3 is "much cleaner and easier to recognize", the MNIST database is constructed by mixing both, to make the training and test set more similar. The training set is composed of 30,000 patterns from SD3 and 30,000 patterns from SD1 and, analogously, the MNIST test set is composed of 5,000 patterns from SD3 and 5,000 patterns from SD1. However, the sets of writers of the training set and test set were disjoint to make the test conditions more realistic [9].

From the point of view of a practical application, it is quite reasonable to verify how robust is the considered method in recognition of previously unseen digits written by different persons. Nevertheless, from the theoretical point of view, the statistical properties of the training and test set should be identical in case of a generally applicable benchmark database. We show that, despite the mixing of SD1 and SD3 numerals, the MNIST training and test-data sets still have slightly different statistical properties (probably due to different writers) which may affect the evaluation of the applied recognition methods. For example, a detailed analysis of misclassified MNIST numerals [12] shows that about 60% of errors on the test patterns might be caused by the lack of similar training samples.

It appears that the frequently used enlargement of MNIST training data sets by suitably distorted patterns actually helps to bridge the differences between the training and test sets. We assume that the massive extension of training data based on external knowledge substitutes some missing training counterparts of the unusual test patterns. In this way a medium-level method could become comparable with a strong knowledge-independent classifier.

Without any doubt, in case of a practical problem we should use as much external knowledge as possible both for the feature extraction and data extension techniques. However, if a comparison of classifiers based on a benchmark experiment should be of a general validity, then the training algorithm should not relay on a specific external knowledge about the data since no such knowledge can be assumed available in a general case.

The classification of hand-written numerals is one of the most traditional problems of pattern recognition. In this respect the MNIST database with its long history is an important benchmark to compare different methods of recognition of handwritten digits. Nonetheless, because of the specific differences between the data sets and widely used external knowledge in the classifier design, its comparative value is more or less restricted to recognition of the MNIST numerals.

Aming at a more general applicability we suggest in this paper an example of a large and easily applicable benchmark based on artificial binary patterns randomly generated on a chess-board by moving the chess-pieces rook and knight. The two classes "ROOK" and "KNIGHT" may overlap and the related classification problem has a clear statistical nature. Except for the two underlying chess-moves there is no external knowledge about the data

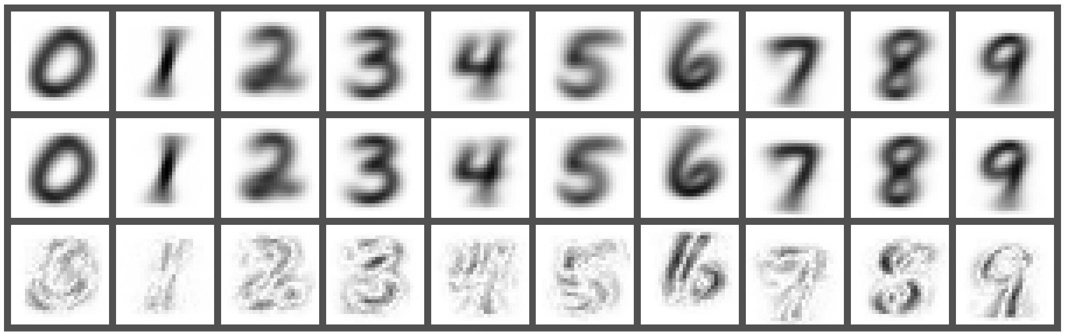
Figure 1. Comparison of “mean numerals” (mean grey levels) from the MNIST training set (first row) and test set (second row). The third row contains absolute differences of both (rescaled 8-times).

and there is no simple way to extract informative features from the binary vectors. The data sets are easily reproducible by using uniquely initialized sequences of pseudo-random numbers.

The paper is organized as follows: In Sec. 2 we discuss some MNIST benchmark results, in Sec. 3 and 4 we recall the basic features of probabilistic neural networks and their implementation aspects. The numerical experiments are described in Sec. 5. and Sec. 6 suggests a novel statistically balanced benchmark. Sec. 7 summarizes the concluding remarks.

2. Comments on Related Work

According to [9], the original black and white (bilevel) NIST images were size normalized to a 20×20 box and then centralized in a 28×28 field. The resulting MNIST images are described in grey levels (0-255) as a result of the normalization algorithm. The different statistical properties of the MNIST training and test data sets are even visible from the comparison of the mean marginal grey levels in Fig.1. We recall that different marginals imply different properties but more complex statistical properties may differ with identical marginals too. By comparing the mean training images to the mean test images we have noticed that the test digits are slightly more upright. This is probably the reason why, in some cases (13 references in [9]), a directly used deskewing or deslanting of numerals helps to improve the classification accuracy.

The most successful methods apply some data enlargement techniques to generate differently modified versions of the original MNIST images (27 references in [9]). In particular, the best classifiers make use of different affine or elastic distortions, shifts, skewing, scaling, compression or even very general nonlinear random transforms in order to extend the training set. According to [1] the deformations of training images are essential to prevent overfitting and greatly improve generalization.

We assume that the distorted numerals help to compensate for the missing training counterparts of the unusual test set images. In this sense, the data extension techniques as well as specific feature extraction methods often utilize the intuitive ”external” knowledge about the semantic invariance of handwritten numerals. The resulting error rates are very low (about 0.5%) with the recent top result of 0.19% [11]. The last paper combines elastic distortion technique with additional scaling and rotation transforms to obtain strongly distorted training images. In the survey paper [10] the benefit of data enlargement methods is not discussed.

In this paper we use the probabilistic neural networks (PNN), cf. [3]–[8], to verify the specific properties of the MNIST data. The PNN as a generative model assume the Bayesian classification according to the posterior probabilities. For this purpose, we approximate the class-conditional distributions directly in the sample space by multivariate Bernoulli mixtures. In this respect the PNN represent one of the few methods to classify the MNIST data by explicit evaluation of the Bayes formula without any feature extraction.

The Bayes decision rule minimizes the error probability provided that the conditional distributions are correctly estimated and therefore, unlike e.g. discriminative classifiers, there is a close relation between the statistical properties of the source data sets and the resulting classification accuracy.

3. Probabilistic Neural Networks

The statistical recognition of handwritten numerals presumes the probabilistic description of classes. Considering the numerals described by means of binary vectors

$$\boldsymbol{x} = (x_1, \dots, x_N) \in \mathcal{X}, \;\; \mathcal{X} = \{0,1\}^N,$$

we approximate the class-conditional distributions $P(\boldsymbol{x}|\omega)$ by multivariate Bernoulli mixtures

$$P(\boldsymbol{x}|\omega) = \sum_{m \in \mathcal{M}_\omega} w_m F(\boldsymbol{x}|m) = \sum_{m \in \mathcal{M}_\omega} w_m \prod_{n \in \mathcal{N}} f_n(x_n|m), \tag{1}$$

$$\sum_{m \in \mathcal{M}_\omega} w_m = 1, \;\; \mathcal{N} = \{1, \dots, N\}, \;\; \bigcup_{\omega \in \Omega} \mathcal{M}_\omega = \mathcal{M}.$$

Here $w_m \geq 0$ are probabilistic weights, $\mathcal{M}_\omega$ are the component index sets, $\mathcal{N}$ is the index set of variables, $F(\boldsymbol{x}|m)$ denote the components and $f_n(x_n|m)$ are the univariate Bernoulli distributions with the parameters θ_{mn}:

$$f_n(x_n|m) = \theta_{mn}^{x_n}(1-\theta_{mn})^{1-x_n}, \;\; n \in \mathcal{N}. \tag{2}$$

The basic idea of probabilistic neural networks is to assume the components (neurons) $F(\boldsymbol{x}|m)$ in the form

$$F(\boldsymbol{x}|m) = F(\boldsymbol{x}|0) G(\boldsymbol{x}|m, \phi_m), \; m \in \mathcal{M}_\omega \tag{3}$$

where $F(\boldsymbol{x}|0)$ is a “background” distribution defined as a fixed product of global unconditional marginals

$$F(\boldsymbol{x}|0) = \prod_{n \in \mathcal{N}} \theta_{0n}^{x_n}(1-\theta_{0n})^{1-x_n}, \quad (\theta_{0n} = \mathcal{P}\{x_n = 1\}) \tag{4}$$

and $G(\boldsymbol{x}|m,\phi_m)$ are the component functions including binary structural parameters $\phi_{mn} \in \{0,1\}$

$$G(\boldsymbol{x}|m,\phi_m) = \prod_{n\in\mathcal{N}} \left[\frac{f_n(x_n|m)}{f_n(x_n|0)}\right]^{\phi_{mn}} = \prod_{n\in\mathcal{N}} \left[\left(\frac{\theta_{mn}}{\theta_{0n}}\right)^{x_n}\left(\frac{1-\theta_{mn}}{1-\theta_{0n}}\right)^{1-x_n}\right]^{\phi_{mn}}. \quad (5)$$

Using the structural mixture model we can cancel the background distribution $F(\boldsymbol{x}|0)$ in the Bayes formula

$$p(\omega|\boldsymbol{x}) = \frac{p(\omega)\sum_{m\in\mathcal{M}_\omega} w_m G(\boldsymbol{x}|m,\phi_m)}{\sum_{\omega\in\Omega} p(\omega)\sum_{j\in\mathcal{M}_\omega} w_j G(\boldsymbol{x}|j,\phi_j)} \quad (6)$$

and therefore the decision making depends on the component functions $G(\cdot|\cdot)$ defined on different subspaces. In other words the input connections of a probabilistic neuron can be confined to any subset of input variables.

The PNN defined by the structural mixture model can be optimized in full generality by means of the EM algorithm (cf. [8, 3, 5]). In particular, given a training set

$$\mathcal{S}_\omega = \{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(K_\omega)}\}, \quad \boldsymbol{x}^{(k)} \in \mathcal{X}, \ \ (\omega \in \Omega),$$

we can estimate the conditional distribution $P(\boldsymbol{x}|\omega)$ by maximizing the log-likelihood function

$$L = \frac{1}{|\mathcal{S}_\omega|}\sum_{x\in\mathcal{S}_\omega} \log\left[\sum_{m\in\mathcal{M}_\omega} w_m F(\boldsymbol{x}|0)G(\boldsymbol{x}|m,\phi_m)\right]. \quad (7)$$

The structural EM algorithm has the form (cf. [8, 5]):

$$q(m|\boldsymbol{x}) = \frac{G(\boldsymbol{x}|m,\phi_m)w_m}{\sum_{j\in\mathcal{M}_\omega} G(\boldsymbol{x}|j,\phi_j)w_j}, \ \boldsymbol{x}\in\mathcal{S}_\omega, \ m\in\mathcal{M}_\omega, \quad (8)$$

$$w'_m = \frac{1}{|\mathcal{S}_\omega|}\sum_{x\in\mathcal{S}_\omega} q(m|\boldsymbol{x}), \quad \theta'_{mn} = \frac{1}{|\mathcal{S}_\omega|w'_m}\sum_{x\in\mathcal{S}_\omega} x_n q(m|\boldsymbol{x}), \quad n\in\mathcal{N}, \quad (9)$$

$$\gamma'_{mn} = \left[\theta'_{mn}\log\frac{\theta'_{mn}}{\theta_{0n}} + (1-\theta'_{mn})\log\frac{(1-\theta'_{mn})}{(1-\theta_{0n})}\right], \quad (10)$$

$$\phi'_{mn} = \begin{cases} 1, & \gamma'_{mn} > \tau_m \\ 0, & \gamma'_{mn} \le \tau_m \end{cases}, \quad \tau_m = \frac{\rho}{N}\sum_{n\in\mathcal{N}} \gamma'_{mn}. \quad (11)$$

Here w'_m, θ'_{mn}, and ϕ'_{mn} are the new iteration values of the mixture parameters, γ'_{mn} are the structural criterion values, τ_m is a component-specific threshold and $\rho \ge 0$ is an optional coefficient to control model complexity. The increasing coefficient ρ reduces the number of parameters in components.

Unlike previous papers [5, 7], we do not optimize the structural properties globally but only at the level of individual neurons. The novel component-specific optimization approach does not suppress the low-weight components (cf. [5, 8]) and is less prone to

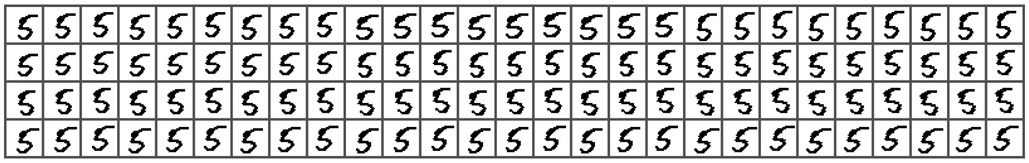

Figure 2. Examples of distorted variants of an original numeral.

overfitting. The iterative equations (8)-(11) generate a nondecreasing sequence $\{L^{(t)}\}_0^\infty$ converging to a (possibly local) maximum of the log-likelihood function (7), (cf. [8]).

The structural criterion γ'_{mn} represents the Kullback-Leibler information divergence $I(f'_n(\cdot|m)|f_n(\cdot|0))$ between the component-specific distribution $f'_n(x_n|m)$ and the corresponding univariate "background" distribution $f_n(x_n|0)$:

$$\gamma'_{mn} = \sum_{\xi=0}^{1} f'_n(\xi|m) \log \frac{f'_n(\xi|m)}{f_n(\xi|0)} = I(f'_n(\cdot|m)|f_n(\cdot|0))$$

and, in this sense, only the most informative conditional distributions $f'_n(\cdot|m)$ are included in the structural mixture model at each iteration.

In a series of papers (cf. [3] - [8]) we have shown that the PNN have many properties well interpretable from the point of view of biological neural networks. Without any preliminary extraction of features we use a structural mixture model to optimize the connections of neurons with the input variables. In this way each neuron can use its own optimal "receptive field" defined by a subset of input features. PNNs can be combined in a parallel and vertical sense, the probabilistic neuron has strictly modular properties and the sequential version of the EM algorithm ([4]) supports the Hebbian principle of learning.

We recall that the structural mixture model is invariant with respect to arbitrary permutation of input variables (cf. [5]), the topological properties of the raster are identified statistically. This is a quite important neuromorphic feature of PNN since e.g. the higher-level neurons in the brain have no information about the topological structure of the retina.

4. Implementation of PNN

For the sake of numerical experiments we have transformed the original nearly binary MNIST variables $\xi_n \in \langle 0, 255 \rangle$ to a strict binary form ($\xi_n > 50 \Rightarrow x_n = 1$). Formally the binarization is accompanied by an information loss but this is well counter-balanced by the resulting efficient description of the data.

The implementation of the EM algorithm (8)-(11) includes two main parameters, namely the number of mixture components $|\mathcal{M}_\omega|$ (identical for all classes $\omega \in \Omega$), and the coefficient ρ to control the number of mixture parameters (model complexity). In all experiments, we have changed the two parameters in relatively wide bounds to illustrate their limited relevance.

The EM algorithm has always been initialized randomly using uniform component weights $w_m = 1/|\mathcal{M}_\omega|, \; m \in \mathcal{M}_\omega$ and by specifying the component parameters θ_{mn}

according to randomly chosen data vectors $\boldsymbol{x} \in \mathcal{S}_\omega$:

$$\theta_{mn} = 0.4x_n + 0.1(1 - x_n), \quad n \in \mathcal{N},\ m \in \mathcal{M}_\omega. \tag{12}$$

A small regularizing constant was added to the estimated parameters to avoid singularities in Eq. (10):

$$\theta^{'}_{mn} = \frac{1}{|\mathcal{S}_\omega| w^{'}_m}[0.01 + \sum_{x \in \mathcal{S}_\omega} x_n q(m|\boldsymbol{x})], \quad n \in \mathcal{N}. \tag{13}$$

The conditional weights $q(m|\boldsymbol{x})$ have to be evaluated in logarithmic form and carefully normalized because of the latent risk of multiple underflow.

The EM iterations have been stopped by the relative increment threshold. We needed about 10–15 iterations to satisfy the stopping rule condition $(L^{'} - L)/|L| < 0.001$. The CPU time (single core of the processor Intel Core i5-4690 CPU@3.50GHz) depends on the number of mixture components $|\mathcal{M}_\omega|$ and the complexity parameter ρ. The time for a complete solution of the problem including estimation of all ten class-conditional distributions from the original data was in tens of minutes. In case of the extended data (117-times) the complete solution needed between 5 hours $(|\mathcal{M}_\omega| = 300, \rho = 0.35)$ and 48 hours $(|\mathcal{M}_\omega| = 4,000, \rho = 0.15)$.

5. Numerical Experiments

5.1. Non-extended Data

The initial computational experiments (Table 1) refer to the original non-extended binarized data. First, we approximated each conditional distribution $P(\boldsymbol{x}|\omega)$ by a single product of 784 univariate marginal distributions (2). The related mean Bayesian classification error obtained for the independent test set is 15.47% (cf. Table 1, experiment I). In the next experiments II – X with the same data we have verified the recognition error of differently specified structural mixtures $(|\mathcal{M}_\omega| = 50 - 1000, \rho = 0.05 - 0.15)$ with the best result 3.94% in the experiment V. There is a clear tendency to overfitting with more than 100 components. The resulting recognition accuracy is starting-point dependent in all tables but, as it can be seen, the corresponding differences are small (in fractions of %). The mean number of utilized parameters θ_{mn} (per component) is given in the third row.

5.2. Extended Data

We assume that the relatively high recognition error of about 4% is partly due to small size of the training set. For this reason we extended the training data by means of simple transforms in the experiments of Table 2. We generated 117 variants for each training numeral by combining simple shifts and adding or removing rows or columns, as illustrated in Fig. 2. In this way we use external knowledge about the natural invariance of hand-written digits but the distortions are rather limited. We recall that the shifting of images has a ”natural” biological counterpart in so called saccadic movements of human eyes.

Unlike similar data extension methods we use the modification of numerals in the classification step too (cf. [5]). In particular, we compute the Bayes probabilities $p(\omega|\boldsymbol{x})$ for all

Experiment:	I	II	III	IV	V	VI	VII	VIII	IX	X
Comps. $\lvert\mathcal{M}_\omega\rvert$:	1	50	80	100	100	100	200	200	500	1000
Parameters	784.0	709.8	693.4	679.4	700.6	676.8	487.6	644.4	535.7	642.4
Error [%]:	**15.47**	**4.28**	**4.19**	**4.12**	**3.94**	**4.18**	**4.22**	**4.51**	**4.56**	**4.66**

Table 1. Recognition of the original non-extended MNIST training data and the independent test data. The table shows the number of components in classes (2nd row), mean number of mixture parameters per component (3rd row) and the resulting mean error rate (last row).

Experiment:	I	II	III	IV	V	VI	VII	VIII	IX	X
Comps. $\lvert\mathcal{M}_\omega\rvert$:	1	300	500	800	1000	1000	1500	2000	3000	4000
Parameters	784.0	454.9	416.2	276.2	373.5	373.1	388.6	416.5	389.3	390.5
Error [%]:	**20.80**	**2.33**	**1.86**	**1.81**	**1.66**	**1.55**	**1.82**	**1.77**	**1.64**	**1.65**

Table 2. Recognition based on the MNIST training set and the independent test set, both extended 117-times by slightly distorted original numerals (cf. Fig.2).

117 variants of each tested numeral $\boldsymbol{x} \in \mathcal{X}$ and use the mean posterior probability for the final classification:

$$\omega_0 = \arg\max_{\omega\in\Omega}\{\bar{p}(\omega|\boldsymbol{x})\}, \;\; \bar{p}(\omega|\boldsymbol{x}) = \frac{1}{117}\sum_{i=1}^{117} p(\omega|\boldsymbol{x}^{(i)}). \tag{14}$$

The decision-making based on the mean posterior probabilities is very similar to majority voting (because of nearly binary values $p(\omega|\boldsymbol{x}^{(i)})$) and reduces the classification error essentially.

The experiments based on the extended data sets are shown in Table 2. The results correspond to differently specified structural mixtures ($\rho = 0.15 - 0.35$) with the best classification error of 1.55% (Table 2, experiment VI). This result corresponds well to our previous experiments with similarly normalized NIST data (cf. [5]) and also to other MNIST results achieved in the sample space without feature extraction, e.g. k-NN rule: 3.66%, RBF-network: 2.53%, multilayer perceptron: 1.91%, support vector classifier: 1.41% (cf. [10], Table 8).

5.3. Test Data Included into Training

At the first view, the best classification results on the MNIST benchmark (cf. [9]) suggest that a suitable extension of the training data by means of distorted patterns is rather important to achieve a low error rate. Considering a general intuitive knowledge about the semantic invariance of hand-written numerals with respect to different deformations we can propose a wide variety of distorting transforms. Moreover, by displaying the misclassified test numerals we can get useful hints regarding how the training numerals should be modified.

Experiment:	I	II	III	IV	V	VI	VII	VIII	IX	X
Comps. $\lvert\mathcal{M}_\omega\rvert$:	500	500	1000	1500	1500	2000	2500	3000	3500	4000
Parameters	738.6	741.2	738.3	739.5	740.7	739.5	740.3	739.3	738.6	739.1
Error [%]:	**0.49**	**0.45**	**0.15**	**0.11**	**0.10**	**0.09**	**0.06**	**0.04**	**0.04**	**0.02**

Table 3. Both non-extended test- and training set used for training. The resulting "positively biased" error rate is decreasing with the increasing mixture complexity.

Experiment:	I	II	III	IV	V	VI	VII	VIII	IX	X
Comps. $\lvert\mathcal{M}_\omega\rvert$:	100	200	300	500	800	100	500	1000	2000	3000
Parameters	352.0	727.0	373.5	438.9	352.0	700.6	759.7	742.8	742.7	742.7
Error [%]:	**0.680**	**0.280**	**0.120**	**0.050**	**0.000**	**1.667**	**0.317**	**0.088**	**0.017**	**0.003**

Table 4. Re-substitution experiments with identical non-extended data used for training and testing. Test data: experiments I-V, training data: experiments VI-X.

Experiment:	I	II	III	IV	V	VI	VII	VIII	IX
Comps. $\lvert\mathcal{M}_\omega\rvert$:	100	500	1000	2000	100	200	500	1000	2000
Error (valid.) [%]	**2.167**	**0.953**	**0.645**	**0.461**	**2.170**	**1.546**	**0.950**	**0.652**	**0.449**
Error (test) [%]	**2.185**	**0.970**	**0.659**	**0.472**	**2.170**	**1.539**	**0.963**	**0.652**	**0.456**

Table 5. Classification of the rook-made and knight-made patterns on the 16×16 chess-board, number of positions NX=25. The table shows the number of components in classes and the resulting mean error rate on the validation-set and test-set (rows 3,4).

From this point of view the "best possible" extension can be obtained by directly including all the test numerals in the training data set. The corresponding "illegal" results of recognition experiments, when the joined (non-extended) training and test sets were used for estimating the class-conditional mixtures $P(\boldsymbol{x}|\omega)$, are shown in Table 3. The best (positively biased) classification error 0.02% (Table 3, experiment X) is no more independent, but it is surprisingly small. This can be seen as an indirect evidence that the statistical properties of the training and test set are different. We assume that in case of sufficiently large and statistically well balanced benchmark the inclusion of test samples into the training set would have little or no influence on the recognition error.

5.4. Resubstitution Experiments

In order to clarify the "illegal" results in the Table 3 we have made so-called re-substitution experiments - with the same (non-extended) data set used both for training and testing. We recall that there is no risk of overfitting in this case since any outliers in the training set and the test set are identical. Consequently, by increasing the number of components the classification error should approach zero.

By using alone the non-extended test data both for training and testing (Table 4, experiments I-V), we obtained the theoretically achievable zero re-substitution error with $|\mathcal{M}_\omega| = 800$ components, (Table 4, experiment V). It is interesting that, by computing analogous re-substitution errors for the non-extended training data, we obtained worse results (Table 4, experiments VI-X). Again, there is no risk of overfitting but even for large number of components the re-substitution error is decreasing slowly (0.003%, experiment X), probably because the training set is six times larger.

We can conclude that there is an indirect evidence of different statistical properties of the training- and test-set numerals. First, the obvious tendency to overfitting in Table 1 suggests that many training numerals do not occur in the test set and vice versa. Second, the classification accuracy improves essentially by including the test numerals into training because of lack of similar data in the original training set (Table 3). Finally, the resubstitution experiments prove reliable classification performance of PNN in terms of theoretically achievable results (Table 4).

6. Statistically Balanced Benchmark

The main goal of pattern recognition theory is to develop general problem-independent methods. In order to prevent the classification benchmark from dependence on the implicit or explicit external knowledge we consider the recognition of artificial binary patterns on a chessboard. For the sake of the benchmark model we generate two classes of binary patterns by random moves of the chess-pieces rook and knight. Unlike previous experiments [6] we use a chessboard of size 16×16, a fix number of 25 generated positions and reproducible sequences of pseudo-random numbers. Thus each pattern is described by a 256-dimensional binary vector containing 25 ones and 231 zeros.

The procedures to generate the rook-made and the knight-made patterns are described in the Appendix. We remark that the procedure KNIGHT generates the pattern in two separate sequences to produce less condensed images on the chess-board. Each data set is uniquely defined by the related procedure and by the initial "seed" of the pseudo-random generator. Thus, the sequence of patterns is exactly reproducible and arbitrarily long. The data vectors need not be stored, because the procedures are very simple and can be used on-line. The application of the benchmark is very simple, because we need only the two procedures in the Appendix and six seeds (cf. Fig. 3) to reproduce two training sets, two validation sets and two test sets, without any necessity to download and decode any data. A correctly generated sequence of patterns can be easily verified by a single pattern with a given sequence number. For this purpose Fig. 3 shows the three pairs of rook-made and knight-made patterns with the sequence number n=1000, for the training sets, validation sets and test sets respectively.

The statistical properties of the two classes "ROOK" and "KNIGHT" depend on the respective procedures and the pseudo-random sequences. We use a popular pseudo-random sequence RANDU defined by the recurrent relation

$$R_{n+1} = (R_n * 65539) \bmod 2147483648, \; R_0 \approx \text{seed}, \tag{15}$$

where the seed can be arbitrary odd number. The sequence of pseudo-random numbers is periodical with a large period (maximally 2147483648), but the periodicity of the generated patterns is probably much larger as long as the next occurrence of the initial seed does not come exactly at the input of the generating procedure. Possible irregularities of the pseudo-random sequence (like correlations between consecutive numbers) are irrelevant in our case because they only introduce a statistical ”color” into the generated patterns. We assume that the statistical properties of pattern sequences differing only by the initial seed will be nearly identical (cf. Tab. 5).

The data sets generated from the two classes ”ROOK” and ”KNIGHT” may overlap and the related classification problem has a clear statistical nature. Except for the two underlying chess-moves there is no external knowledge about the data and there is no simple way to extract informative features from the binary vectors.

Table 5 shows the results of numerical experiments with the artificial chess-board benchmark. We have verified the classification accuracy for differently complex conditional distributions (cf. the number of mixture components in the second row). The next two rows correspond to independent validation- and test-sets respectively. In the first group of experiments (col. I – IV) the size of all data sets was $|\mathcal{S}_R| = |\mathcal{S}_K| = 10^6$ patterns and in the second group (col. V – IX) twice greater, i.e. $|\mathcal{S}_R| = |\mathcal{S}_K| = 2 * 10^6$ patterns. In view of the large data sets we have estimated the Bernoulli mixtures in full version including all parameters, without structural modification.

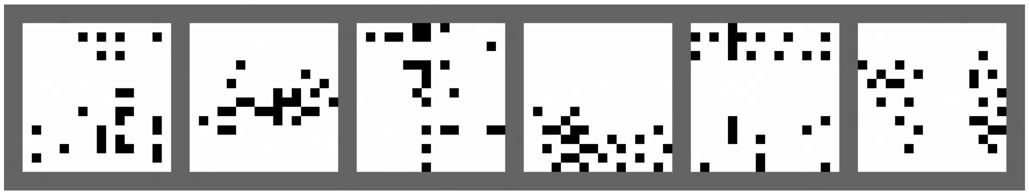

Figure 3. Rook- and knight-patterns with the sequence number $t = 1000$ for training set, validation set and test set. See Appendix for the corresponding procedures ROOK and KNIGHT.

It can be seen that the results obtained for validation- and test-data are very similar because the statistical properties of the corresponding data sets are nearly identical. In both groups the error rates decrease with the increasing complexity, but the corresponding results are not very different despite the twice larger data sets in the second group of experiments. We can conclude, that the number of training patterns in the first group is already large enough to get reliable estimates of the conditional distributions. The best error rate 0.449% (col. IX) is probably near to the underlying theoretically achievable value.

7. Conclusions

The recognition of MNIST numerals is negatively influenced by the slightly different statistical properties of the training and test-sets. We assume that the widely used approach of extending the training data by randomly distorted numerals actually helps to overcome these differences by generating missing variants of the atypical test set numerals. In this sense the application of MNIST benchmark may depend on the external knowledge relating to the semantic invariance of hand-written digits and therefore the comparative value of the benchmark partly confines to MNIST numerals.

With the aim of a more general applicability we suggest a model of a perfectly balanced statistical benchmark based on artificial binary patterns generated on a chess-board by random moves of the chess-pieces rook and knight. By using uniquely initialized pseudo-random sequences the data sets are exactly reproducible in arbitrary length and therefore any data enlargement is unnecessary. There is no external knowledge about the binary patterns except for generating rules and there is no simple way to extract informative features from the data. Obviously, we could create in a similar way a benchmark based on real data vectors or color images.

8. Appendix

Procedure ROOK

```
const int  RS=16, RS2=8, NPOINT=25;
unsigned long  RNR,RNK,CM=2147483648;  unsigned char  X[NN],XB[RS][RS];

int ROOK() // ROOK-pattern synthesis
{//initial seeds: training RNR=1987654327,
   validation RNR=2987654327, testing RNR=3987654327
   int I,J,K,I0,J0,NX;
   for(I=0;I<RS;I++) for(J=0;J<RS;J++) XB[I][J]=0;
   RNR=(65539*RNR)%CM; I0=(RNR/10000)%RS;
   RNR=(65539*RNR)%CM; J0=(RNR/10000)%RS;
   XB[I0][J0]=1;  NX=1;
   while(NX<NPOINT)
   {   RNR=(65539*RNR)%CM; J=(RNR/10000)%RS;
       RNR=(65539*RNR)%CM; K=(RNR/10000)%RS;
       if(J<RS2) I0=K; else J0=K; if(XB[I0][J0]<1){XB[I0][J0]=1;NX++;}
   }   // end of while-loop
   NX=0;  for(I=0;I<RS;I++)  for(J=0;J<RS;J++) X[NX++]=XB[I][J];
   return 0;
}  // end of ROOK
```

Procedure KNIGHT

```
int KNIGHT() // KNIGHT-pattern synthesis
{//initial seeds: training RNK=1987654325,
   validation RNK=2987654325, testing RNK=3987654325
   int I,J,K,I0,J0,NX;
   for(I=0;I<RS;I++) for(J=0;J<RS;J++) XB[I][J]=0;
   RNK=(65539*RNK)%CM;I0=(RNK/10000)%RS;
   RNK=(65539*RNK)%CM;J0=(RNK/10000)%RS;
   XB[I0][J0]=1;  NX=1;
   while(NX<NPOINT)
   {  if(NX==NP2)
      { RNK=(65539*RNK)%CM; I0=(RNK/10000)%RS;
      RNK=(65539*RNK)%CM; J0=(RNK/10000)%RS;}
      RNK=(65539*RNK)%CM; K=(RNK/10000)%RS2;
      switch (K)
      {  case 0: I=I0+2; J=J0+1; break;  case 1: I=I0+1; J=J0+2; break;
         case 2: I=I0+2; J=J0-1; break;  case 3: I=I0+1; J=J0-2; break;
         case 4: I=I0-2; J=J0+1; break;  case 5: I=I0-1; J=J0+2; break;
         case 6: I=I0-2; J=J0-1; break;  case 7: I=I0-1; J=J0-2; break;
      }  // end of switch
      if(I>-1)if(J>-1)if(I<RS)if(J<RS)
      {  I0=I; J0=J; if(XB[I0][J0]<1){XB[I0][J0]=1; NX++;}
      }  // end of if-clause
} // end of while-loop
    NX=0; for(I=0;I<RS;I++) for(J=0;J<RS;J++) X[NX++]=XB[I][J];
    return 0;
}  // end of KNIGHT
```

9. Acknowledgement

This project is supported by the Grant No. 17-18407S of the Czech Science Foundation.

References

[1] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber: Convolutional neural network committees for handwritten character classification. In: Proc. 2011 International Conference on Document Analysis and Recognition (ICDAR), (2011) 1135–1139.

[2] Coop, R., Mishtal, A., I. Arel: Ensemble Learning In: Fixed Expansion Layer Networks for Mitigating Catastrophic Forgetting. IEEE Trans. on Neural Networks and Learning Systems, 24(10), (2013) 1623–1634.

[3] J. Grim: Neuromorphic features of probabilistic neural networks, Kybernetika, 43(5), (2007) 697–712. *http://dml.cz/dmlcz/135807*

[4] J. Grim: A sequential modification of EM algorithm, In: Studies in Classification, Data Analysis and Knowledge Organization, Gaul W., Locarek-Junge H., (Eds.), Springer, (1999), 163 - 170.

[5] J. Grim, J. Hora: Iterative principles of recognition in probabilistic neural networks, Neural Networks, 21(6), (2008) 838–846.

[6] J. Grim and J. Hora: Computational Properties of Probabilistic Neural Networks, Artificial Neural Networks - ICANN 2010 Part II, Springer: Berlin, LNCS **5164**, (2010) 52–61.

[7] J. Grim, J. Kittler, P. Pudil, P. Somol: Multiple classifier fusion in probabilistic neural networks, Pattern Analysis and Applications, 5(7), (2002) 221–233.

[8] J. Grim, P. Pudil, P. Somol: Recognition of handwritten numerals by structural probabilistic neural networks. In: Proceedings of the Second ICSC Symposium on Neural Computation. (Bothe, H., Rojas, R. eds.). ICSC, Wetaskiwin, (2000) 528–534.

[9] Y. LeCun, C. Cortes, C.J.C Burges: The MNIST Database of Handwritten Digits, http://yann.lecun.com/exdb/mnist/ (2013).

[10] C.L. Liu, K. Nakashima, H. Sako, and H. Fujisawa: Handwritten digit recognition: benchmarking of state-of-the-art techniques. Pattern Recognition, 36(10), (2003) 2271–2285.

[11] X.X. Niu, and C.Y. Suen: A novel hybrid CNNSVM classifier for recognizing handwritten digits. Pattern Recognition, 45(4), (2012) 1318–1325.

[12] C.Y. Suen, and J. Tan: Analysis of errors of handwritten digits made by a multitude of classifiers. Pattern Recognition Letters, 26(3), (2005) 369–379.