An Interval Approach to Pattern Recognition of Numerical Matrices*

Alexander V. Prolubnikov

Omsk State University, Omsk, Russia

a.v.prolubnikov@mail.ru

Abstract

We present an interval approach for pattern recognition of numerical matrices. We construct systems of interval linear equations that are associated with given numerical pattern matrices. Considering a system of interval linear equations as a family of systems of real linear equations, we use a measure of variation of these systems solutions as a measure of distance between matrices. As an application, we use our technique for the recognition of raster images that are distorted in the course of noising. The results of computational experiments are presented.

Keywords: pattern recognition, interval analysis
AMS subject classifications: 68T10

1 Introduction

Registration of any data by technical means often is complicated by noise (e.g., measurement errors) that interferes with the registration process. A common problem is the recognition of patterns under specified constraints on the noise. Considering the data presented in a matrix form, we are given a set of pattern matrices and a matrix obtained from some pattern matrix in the course of noising. We need to identify this pattern matrix. One commercially significant application is registration of raster images, for example, in medical applications.

Currently existing recognition methods for such problems may be divided into two classes: those using a preliminary learning stage, and those that do not. Learning implies that a processing unit is capable of changing its input/output behavior as a result of changes on the input. Algorithms based on the theory of morphological analysis, Kora-type algorithms, and neural network-based algorithms are some examples of methods with a learning stage. Examples of the second class of methods include the nearest neighbor method, the k nearest neighbor method, the potential function method, and the Parzen window method.

*Submitted: December 30, 2012; Revised: October 10, 2013; Accepted: November 14, 2013.

We propose an original measure of closeness (distance) between matrices and a pattern recognition algorithm with no learning stage. For given input matrices, the algorithm constructs special systems of interval linear equations. Considering an interval system of equations as a set of point systems of equations, we perform recognition by minimizing a measure of the variation of the solutions to point systems. To evaluate the variation, we use the Lebesgue measure of an enclosure of the solution set to the interval linear system.

2 Preliminaries

Throughout the paper, we use the following notation and definitions.

Boldface font is used for intervals, interval vectors and interval matrices. A special font is used for input data matrices: we denote the k-th pattern matrix as $\mathsf{A}^{(k)}$ and the matrix to be recognized as A, $\mathsf{a}_{ij}^{(k)}$, and let a_{ij} denote elements of these matrices.

For $A \in \mathbb{R}^{m\times n}$, $A = (a_{ij})$, we use the p-norm $\|A\|_p$:

$$\|A\|_p = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^p\right)^{1/p},$$

$p \in \mathbb{R}$, $p \geq 1$. Also, we use the vector norm $\|x\|_\infty$ for $x \in \mathbb{R}^n$:

$$\|x\|_\infty = \max_{1\leq i\leq n} |x_i|.$$

For $A \in \mathbb{R}^{m\times n}$, the matrix norm induced by $\|\cdot\|_\infty$ is denoted by

$$\|A\|_\infty = \max_{1\leq i\leq m}\left(\sum_{j=1}^{n}|a_{ij}|\right).$$

Definition 1. Let $\boldsymbol{A}x = \boldsymbol{b}$ be an interval linear system of equations, $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$, $\boldsymbol{b} \in \mathbb{IR}^n$. Then the *solution set* of this system is the set

$$\varXi(\boldsymbol{A},\boldsymbol{b}) = \bigl\{\, x \in \mathbb{R}^n \bigm| (\exists A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})\,(Ax = b)\bigr\}.$$

The solution set to interval systems of equations can be defined in a variety of ways, and Definition 1 is only one of them [2]. However, we do not use these solution sets in our work, and the abbreviated term “solution set” should not confuse readers.

Definition 2. A *comparison matrix* of a matrix $A \in \mathbb{R}^{n\times n}$ is a matrix $\langle A\rangle \in \mathbb{R}^{n\times n}$ with the elements

$$\langle A\rangle_{ij} = \begin{cases} |a_{ij}|, & \text{if } i = j,\\ -|a_{ij}|, & \text{if } i \neq j.\end{cases}$$

Definition 3. An *M-matrix* is a matrix $A \in \mathbb{R}^{n\times n}$ representable as $A = sI - P$, for a non-negative matrix P with spectral radius $\rho(P) < s$. A matrix $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ is an *interval M-matrix* if every real matrix $A \in \boldsymbol{A}$ is an M-matrix.

Definition 4. An *H-matrix* is a matrix $A \in \mathbb{R}^{n\times n}$ such that its comparison matrix is an M-matrix. An *interval H-matrix* is a matrix $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ such that every real matrix $A \in \boldsymbol{A}$ is an H-matrix.

3 Recognition of Numerical Matrices Using Lebesgue Measure of the Solution Sets

3.1 General Idea

The problem under study is formulated as follows. We are given a set of N $n \times n$-matrices $\mathsf{A}^{(1)}, \ldots, \mathsf{A}^{(N)} \in \mathbb{R}^{n\times n}$. The matrix $\mathsf{A} = (\mathsf{a}_{ij})$ is obtained from some matrix $\mathsf{A}^{(p)} = (\mathsf{a}_{ij}^{(p)})$ in the course of noising, $p \in \{1, \ldots, N\}$. It is known that values of elements of $\mathsf{A}^{(p)}$ may vary within the intervals $[\mathsf{a}_{ij}^{(p)} - \Delta, \mathsf{a}_{ij}^{(p)} + \Delta]$, $\Delta \geq 0$ $(i, j = \overline{1, n})$. We need to identify p.

Without loss of generality, we may assume that the matrices $\mathsf{A}^{(1)}, \ldots, \mathsf{A}^{(N)}, \mathsf{A}$ are square. In the other cases, if the matrices are $m \times n$-matrices and $m < n$, we append $n - m$ zero rows to every input matrix. If $n < m$, we append $m - n$ zero columns.

To solve the recognition problem, we need a special measure of closeness (distance) between matrices. Suppose we have two matrices A and $B \in \mathbb{R}^{n\times n}$, $A = (a_{ij})$, and $B = (b_{ij})$. Let us construct an interval matrix $\boldsymbol{C}$ from A and B so that the elements $(\boldsymbol{C})_{ij}$ of the matrix $\boldsymbol{C}$ are the intervals

$$(\boldsymbol{C})_{ij} = [\min\{a_{ij}, b_{ij}\}, \max\{a_{ij}, b_{ij}\}]\,. \tag{1}$$

These intervals specify the variations of the matrix B elements that are needed to obtain the matrix A by modifying elements of B.

Assuming that A is obtained from B by adding noise, we measure aggregate variation of the matrix B elements by using the Lebesgue measure of the solution set $\varXi(\boldsymbol{C}, b)$ for some right-hand side vector $b \in \mathbb{R}^n$. The Lebesgue measure of this set depends on a mutual disposition of the matrix elements, and it depends continuously on their changes. We use the Lebesgue measure $\mu(\varXi(\boldsymbol{C}, b))$ as a measure of closeness $\delta_{\varXi}(A, B)$ between the matrices A and B:

$$\delta_{\varXi}(A, B) = \mu\big(\varXi(\boldsymbol{C}, b)\big).$$

The smaller the value of $\delta_{\varXi}(A, B)$, the closer A is to B.

If A is the matrix that was obtained from the pattern matrix $\mathsf{A}^{(p)}$ by adding noise, and if elements of the matrix $\mathsf{A}^{(p)}$ have been varied in some restricted intervals during the process, then the value of the measure may be small enough to recognize $\mathsf{A}^{(p)}$ from the other candidates. Assuming this, we use the heuristic

$$p = \arg\min_{k} \delta_{\varXi}\left(\mathsf{A}, \mathsf{A}^{(k)}\right) \tag{2}$$

for pattern recognition.

The set $\varXi = \varXi(\boldsymbol{C}, b)$ is a union of not more than 2^n polyhedrons formed by intersections of the set $\varXi$ with the orthants of $\mathbb{R}^n$ [1]. The problem of describing this set has an exponential computational complexity by itself, so the computation of the Lebesgue measure of $\varXi$ has an exponential complexity, too. Therefore, we just estimate the Lebesgue measure of $\varXi$. We do this by computing the Lebesgue measure of some approximation of the interval hull $\Box\varXi$ of the solution set. An interval hull $\Box\varXi$ is the smallest, by inclusion, interval box that contains $\varXi$, i. e. $\Box\varXi \subseteq \boldsymbol{W}$ for every box $\boldsymbol{W}$ such that $\varXi \subseteq \boldsymbol{W}$.

Let $\boldsymbol{X} = ([\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n])^\top$ be an outer approximation (enclosure) of $\Box\varXi$ obtained using some interval algorithm *Encl*. $\boldsymbol{X}$ is such a box that $\Box\varXi \subseteq \boldsymbol{X}$. The

Lebesgue measure μ of $\boldsymbol{X}$ is computed as

$$\mu(\boldsymbol{X}) = (\overline{x}_1 - \underline{x}_1) \cdot \ldots \cdot (\overline{x}_n - \underline{x}_n).$$

For every $k \in \{1, \ldots, N\}$, we construct the matrices $\boldsymbol{C}^{(k)}$ according to (1) and take A as A and $\mathsf{A}^{(k)}$ as B. The following natural suggestion is a basis for our novel pattern recognition algorithm: for a fixed right-hand side vector $b \in \mathbb{R}^n$, the smaller the variation of solutions of the real systems that give $\boldsymbol{C}^{(k)}x = b$, the more likely it is that A was obtained from the pattern matrix $\mathsf{A}^{(k)}$.

Our algorithm can be classified as a "nearest neighbor" algorithm that uses measure of closeness δ_{Ξ} as a distance between the matrices. Denoting an approximation (enclosure) of $\Box\Xi^{(k)}$ as $\boldsymbol{X}^{(k)}$, we can formulate our pattern recognition algorithm:

Algorithm for recognition of numerical matrices

Input: pattern matrices $\mathsf{A}^{(1)}, \ldots, \mathsf{A}^{(N)}$ and a matrix A obtained from a pattern matrix by addition of noise.

Output: an index p of one of the pattern matrices, as a result of recognition.

1. For $\mathsf{A}^{(k)}$ and A, compute $\boldsymbol{C}^{(k)}$ according to (1), $k = \overline{1, N}$.
2. Using an enclosure method *Encl*, compute $\boldsymbol{X}^{(k)}$, $k = \overline{1, N}$.
3. Find p such that $p = \arg\min\limits_{k} \mu\left(\boldsymbol{X}^{(k)}\right)$.

3.2 Modification of the Matrices $\boldsymbol{C}^{(k)}$

To perform efficient patter recognition, we preprocess the input matrices.

In the first modification, we increase every element of every input matrix by the same value υ ($\upsilon > 0$):

$$\mathsf{a}_{ij} := \mathsf{a}_{ij} + \upsilon,\ \mathsf{a}_{ij}^{(k)} := \mathsf{a}_{ij}^{(k)} + \upsilon, \quad \text{for } k = \overline{1, N}. \tag{3}$$

The transformations (3) preserve the distance between elements in different positions from a single matrix and preserve the distance between elements in the same positions from two different matrices. Thus, the transformations (3) preserve all of the input information that we may use for recognition purposes.

As the result of the transformations (3), we decrease the ratio $\Delta / \left|\mathsf{a}_{ij}^{(k)}\right|$ for elements of the matrices. Namely, the ratio

$$\Delta / \left|\mathsf{a}_{ij}^{(k)} + \upsilon\right|$$

decreases with the growth of υ. Such a decrease is necessary for the following reasons. If the value Δ is greater than or equal to the absolute values of the elements of the pattern matrices, then for $C \in \boldsymbol{C}^{(k)}$, the vectors $x = C^{-1}e$ may differ so much that we cannot arrive at any definite decision analysing the distances between them. However, if the radius Δ is small enough with respect to moduli of the pattern matrices elements, we often are able to make such a decision.

For example, for matrices of black-and-white images, white pixels are encoded as 1's, and black pixels are encoded as 0's. If some pixels of the pattern image are inverted by the addition of noise, then Δ is greater or equal than every element of the input matrices, and we cannot perform recognition using our heuristic. However, if we modify the input matrices according to (3) using $\upsilon = 10$, which results in $\Delta/|\mathsf{a}_{ij}^{(k)} + \upsilon| \leq 0.1$, we achieve an efficient recognition.

The transformations (3) of the input matrices are equivalent to the transformations of the interval matrices $\boldsymbol{C}^{(k)}$,

$$\boldsymbol{C}^{(k)} := \boldsymbol{C}^{(k)} + \upsilon\boldsymbol{E}, \tag{4}$$

where $\boldsymbol{E}$ is a matrix with the elements $(\boldsymbol{E})_{ij} = [1, 1]$, $i, j = \overline{1, n}$.

In the second modification, we achieve the diagonal dominance in the matrices $\boldsymbol{C}^{(k)}$. The property of diagonal dominance is known to be necessary for the successful execution of many interval methods that compute enclosures of the solution sets to interval linear systems, i. e. for high quality of the boxes $\boldsymbol{X}^{(k)}$ in our case. In particular, the interval Gauss-Seidel iteration works efficiently for interval linear systems with diagonally dominant matrices.

Let $\boldsymbol{D} \in \mathbb{IR}^{n\times n}$ be a diagonal matrix with elements $(\boldsymbol{D})_{ii} = [d, d]$, $i = \overline{1, n}$, where

$$d = 2\max\left\{\max_{i=\overline{1,n}}\left(\sum_{j\neq i}|\mathsf{a}_{ij}|\right), \max_{i=\overline{1,n}}\left(\sum_{j\neq i}|\mathsf{a}_{ij}^{(1)}|\right), \;\ldots,\; \max_{i=\overline{1,n}}\left(\sum_{j\neq i}|\mathsf{a}_{ij}^{(N)}|\right)\right\}.$$

Taking such d, we modify the matrices $\boldsymbol{C}^{(k)}$:

$$\boldsymbol{C}^{(k)} := \boldsymbol{C}^{(k)} + \boldsymbol{D}. \tag{5}$$

3.3 Choosing Right-hand Side Vectors in the Interval Linear Systems

For the interval system of equations $\boldsymbol{C}x = \boldsymbol{b}$, the right-hand side $\boldsymbol{b}$ is a vector that defines the set $\varXi(\boldsymbol{C}, \boldsymbol{b})$ for a given matrix $\boldsymbol{C}$. The choice of $\boldsymbol{b}$ is quite important for the overall efficiency of the algorithm developed.

As the right-hand side vector $\boldsymbol{b}$, we choose a point (non-interval) real vector, since it does not bear any uncertainty in the practical applications of our problem. Such a selection of $\boldsymbol{b}$ allows us to obtain more precise enclosures of the solution sets, because it decreases the distance between $\varXi(\boldsymbol{C}, \boldsymbol{b})$ and $\square\varXi(\boldsymbol{C}, \boldsymbol{b})$ [2].

If the right-hand side $\boldsymbol{b}$ is a point vector b, then

$$\varXi(\boldsymbol{C}, b) = \left\{\, x \in \mathbb{R}^n \;\middle|\; (\exists C \in \boldsymbol{C})\,(Cx = b) \,\right\}.$$

The transformation (5) leads to the matrices $\boldsymbol{C}^{(k)}$ being regular, i.e., every $C \in \boldsymbol{C}^{(k)}$ is a regular (invertible) matrix. For a regular matrix $\boldsymbol{C}$, we have

$$\varXi(\boldsymbol{C}, b) = \left\{\, C^{-1}b \;\middle|\; (C \in \boldsymbol{C}) \,\right\}.$$

The set $\varXi(\boldsymbol{C}, b) \subset \mathbb{R}^n$ can be considered as the image of the set of real matrices $\boldsymbol{C} \subset \mathbb{R}^{n\times n}$. More precisely, $\varXi(\boldsymbol{C}, b)$ is the image of $\boldsymbol{C}$ under such mapping L_b that

$$\begin{aligned} &L_b : \mathbb{R}^{n\times n} \to \mathbb{R}^n, \\ &L_b(C) = C^{-1}b \qquad \text{for every } C \in \boldsymbol{C}. \end{aligned}$$

If we denote the elements of C^{-1} by σ_{ij}, for the i-th component of any $x \in \Xi(\boldsymbol{C}, b)$,

$$x_i = \sum_{j=1}^{n} \sigma_{ij} b_j = \sum_{j=1}^{n} \biggl(\frac{C_{ji}}{\det C} \biggr) b_j,$$

where $C \in \boldsymbol{C}$ satisfies $x = C^{-1}b$, and C_{ji} are cofactors of the elements of C. Thus, the components of the vector b are weights of the columns of C^{-1} that we use to obtain the components of x.

The above reasoning gives yet another condition on the right-hand side of the interval linear systems to be constructed: all the components of the right-hand side must be equal to each other. If we use the vector $e = (1, \ldots, 1)^{\top}$ as the right-hand side vector, then the components of x are equally dependent on all of the cofactors of C, i.e., these components are equally dependent on every element of the matrix.

This is not valid if we choose a vector with non-equal components as the right-hand side vector. For example, if the unit vector $e_i = (0, \ldots, 1, \ldots, 0)^{\top}$ of the standard basis in $\mathbb{R}^n$ (having 1 at the i-th place only) is chosen for this purpose, then

$$x = \frac{1}{\det C} \cdot \Bigl(C_{1i}, \ldots, C_{ni} \Bigr)$$

for $x \in \Xi(\boldsymbol{C}, e_i)$, $x = L_{e_i}(C)$. In this case, elements of the i-th column of C do not affect the values $C_{1i}, \ldots,\ C_{ni}$. As the result, the components of x are not equally dependent on every element of the matrix C.

The elements of C are specified by elements of the input matrices. We proceed from the presupposition that all the elements of the input matrices should be equally taken into account in the process of pattern recognition. This is why we consider the systems of interval linear equations of the form $\boldsymbol{C}^{(k)}x = e$.

3.4 Estimation of the Solution Sets

For the matrix $\boldsymbol{C}^{(k)}$ obtained from (1) and subject to transformations (4) and (5), the comparison matrix $\langle \boldsymbol{C}^{(k)} \rangle$ is an M-matrix, i.e., $\boldsymbol{C}^{(k)}$ itself is an H-matrix. If we use the interval Gauss-Seidel method to compute $\boldsymbol{X}^{(k)}$, then every sufficiently large box is improved by iteration [3]. Thus, taking a sufficiently large box as an initial approximation, we may compute the enclosures $\boldsymbol{X}^{(k)}$ of the sets $\Xi^{(k)}$ with reasonably good quality. For this purpose, we take the box $\boldsymbol{W} = ([-w, w], \ldots, [-w, w])^{\top} \in \mathbb{IR}^n$ which contains all of the solutions sets $\Xi^{(k)}$ for some $w > 0$.

Without loss of generality, we may assume that all elements of every matrix $C \in \boldsymbol{C}^{(k)}$ are positive. Otherwise, we may perform the preliminary transformations (4) with $\upsilon_0 > 0$ such that υ_0 is greater than the modulus of every negative element of A and $\mathsf{A}^{(k)}$ for $k = \overline{1, n}$. From (5), every $C \in \boldsymbol{C}^{(k)}$ is diagonally dominant. For $C \in\ \boldsymbol{C}^{(k)}$, let

$$R_i(C) = c_{ii} - \sum_{j \neq i} c_{ij}, \qquad R_*(C) = \min_{1 \leq i \leq n} R_i(C).$$

We have from [4]

$$\|C^{-1}\|_{\infty} \leq \frac{1}{R_*(C)}.$$

By virtue of (4) and (5), we have $R_*(C) \geq (n-1)\upsilon$, so

$$\|C^{-1}\|_{\infty} \leq \frac{1}{(n-1)\upsilon}.$$

If $x = C^{-1}e \in \varXi^{(k)}$, then

$$\|x\|_\infty \leq \|C^{-1}\|_\infty \|e\|_\infty,$$

i.e.,

$$\|x\|_\infty \leq \frac{1}{(n-1)\upsilon}. \tag{6}$$

The right-hand side of inequality (6) suggests how to construct an initial enclosure $\boldsymbol{W}$ for the solution set. Specifically, we take $w = 1/((n-1)\upsilon)$ for the starting box $\boldsymbol{W}$ of the interval Gauss-Seidel iteration.

As it follows from (6), a large growth of υ may lead to an inappropriate decrease of values $\|x\|_\infty$ for $x \in \varXi^{(k)}$. As the result of such decrease, we may have a situation in which the Lebesgue measures of the enclosures $\boldsymbol{X}^{(k)}$ do not reflect any specificity of the pattern matrices. As the diagonal dominance becomes excessively large, the values of $\mu(\varXi^{(k)})$ may become too small. If this happens, the comparison of the values $\boldsymbol{X}^{(k)}$ will not lead to successful recognition. Such decrease is not admissible due to computational errors of the enclosure methods we may use. For efficient recognition, we need a rather large deviation of the value $\mu(\boldsymbol{X}^{(p)})$ from the other values $\mu(\boldsymbol{X}^{(k)})$. When we perform our computational experiments, we take $\upsilon = 10\,a$, where $a = \max |\mathsf{a}_{ij}^{(k)}|$, $i, j = \overline{1, n}$, $k = \overline{1, N}$.

3.5 Computational Complexity

The complexity of our algorithm depends crucially on the enclosure method used. It is equal to

$$O(N \cdot \mathrm{cmpl}(\mathit{Encl}, n)),$$

where $\mathrm{cmpl}(\mathit{Encl}, n)$ is a computational complexity of the algorithm *Encl*, which we use to obtain the enclosures $\boldsymbol{X}^{(k)}$. If $\mathrm{cmpl}(\mathit{Encl}, n) = O(n^2)$, then the algorithm has the least complexity order among the algorithms that may be designed for the solution of the problem under consideration. More precisely, the computational complexity of any algorithm for the solution of the problem is not less than $O(n^2)$, since every procedure that involves processesing elements of an $n \times n$-matrix has overall complexity not less than $O(n^2)$.

An important particular case is the use the interval Gauss-Seidel method, which we denote as GS. Then the computational complexity of the recognition algorithm is

$$O\bigl(N \cdot \mathrm{cmpl}(GS, n)\bigr) = O(N \cdot N_{GS} \cdot n^2),$$

where N_{GS} is a number of the interval Gauss-Seidel iterations we perform. We take $N_{GS} = 20$ for our computational experiments considered in the next section.

4 Computational Experiments

In the first two computational experiments considered below, we investigate the heuristic efficiency in an application to recognize images of digits. The images we use are presented in black-and-white and grayscale modes with resolutions of 20×20, 35×35, 50×50, and 100×100 pixels. We have done our experiments for Times New Roman, Arial, and Courier New fonts and for the font presented in Figure 1. As the experiments show, the last font is the most difficult for recognition (see below). We use it in the first and in the second experiment we have performed.

The element $\mathsf{a}_{ij}^{(k)}$ of a pattern image matrix $\mathsf{A}^{(k)}$ may take one of the two values:

$$\mathsf{a}_{ij}^{(k)} = \begin{cases} c_1, & \text{if pixel at } ij \text{ position is white,} \\ c_2, & \text{if pixel at } ij \text{ position is black.} \end{cases}$$

If the images are black-and-white, then $c_1 = 1$ and $c_2 = 0$. If they are grayscale, then c_1 and c_2 may take any two values in the range of 0 to 255.

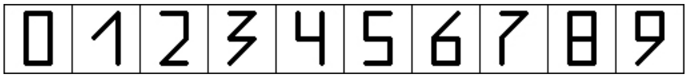

Figure 1: Pattern images of digits.

Suppose we have black-and-white images. Let us take some $Q \in [0, 100]$ as a noise level, i.e., the value of Q specifies a percentage of the pixels subject to noise. We add noise to the pattern image by the following modification of its pixels. For every pixel, we generate a random integer value $q \in [0, 100]$ using a uniform distribution. If $q \in [0, Q]$, then we invert pixel, otherwise, the pixel stays the same. If we add noise to the image in such a manner for $Q = 0$, then we have the initial image. If $Q = 50$, then on average, 50% of the image pixels are inverted. If $Q = 100\%$, then every pixel is inverted in the course of adding noise. For every ordered pair of the pattern matrices $(\mathsf{A}^{(i)}, \mathsf{A}^{(j)})$, we get A by adding noise to $\mathsf{A}^{(i)}$, and then we try to recognize it. During the experiment, we perform 100 trials for every such pair of the matrices. The *recognition efficiency* P is a percentage of right choices that we make,

$$P = \frac{\text{the number of right choices}}{\text{the number of trials}} \times 100.$$

If the images are presented in a grayscale mode, we change the value of any pixel in a predefined interval instead of inverting it.

For the algorithm presented above, results of the computational experiments are shown in Tables 1 and 2 and are illustrated by the graphs in Figure 2. Table 1 shows the results of the computations for $Q \in [31, 50]$. Table 2 shows the results of the computations for $Q = 45\%$ and $n = 50$. If $Q \leq 30\%$, then the recognition efficiency is not less than 99.9%. Graphs in Figure 2 show that the recognition efficiency grows as the image resolution grows.

The pattern recognition efficiency of our algorithm is not worse than that for the other well-known algorithms. For the Kora, R-method, TEMP, and CORAL algorithms, $P = 80\%$ when $Q = 43\%$, the algorithm based on monochrome morphology has $P = 80\%$ when $Q = 45\%$ [5, 6]. For neural network algorithms, the recognition efficiency P is not greater than 90% when $Q \geq 46\%$ for a similar experiment with letter images [7] that we also have performed. If we use an appropriate resolution, our heuristic gives equal or greater recognition efficiency.

We have compared the recognition efficiency of the heuristic (2) with two natural heuristics frequently used for pattern recognition of numerical matrices,

$$p = \arg\min_k \rho_1 \left(\mathsf{A}, \mathsf{A}^{(k)}\right) \tag{7}$$

and

$$p = \arg\min_k \rho_2 \left(\mathsf{A}, \mathsf{A}^{(k)}\right), \tag{8}$$

where $\rho_1 = \|\mathsf{A} - \mathsf{A}^{(k)}\|_1$, and $\rho_2 = \|\mathsf{A} - \mathsf{A}^{(k)}\|_2$. First, consider the heuristic (8).

Table 1: The recognition efficiency P for the test images ($Q \in [31, 50]$).

$n \backslash Q$	31	32	33	34	35	36	37	38	39	40
20	99.6	99.6	99.4	99.2	99	98.6	98	97.3	96	95
35	99.9	100	100	99.8	99.8	99.7	99.5	99.4	99	98.3
50	100	100	100	100	100	100	99.9	99.9	99.8	99.7
100	100	100	100	100	100	100	100	100	100	100
$n \backslash Q$	41	42	43	44	45	46	47	48	49	50
20	93.3	90.7	88.1	84.1	79.9	75.3	69.2	63.1	57	49.7
35	97.5	96.6	94.5	91	87.2	82.2	76.2	67.9	60.1	49.4
50	99.5	99.3	98.4	97.3	95.2	90.6	84	74.6	62.7	49.9
100	100	100	100	99.9	99.7	99.1	97.2	99.1	75.1	50.7

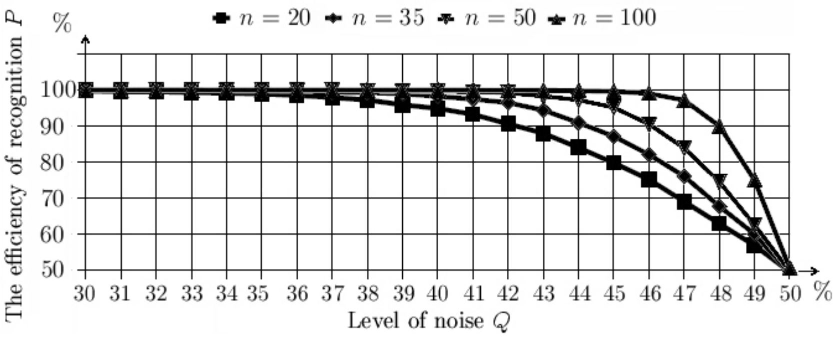

Figure 2: The growth of recognition efficiency for growing image resolution.

Let S be a percentage of the recognition problems for which the heuristic (8) does not give the right solution, while it can be obtained by using the heuristic (2). It may be seen that if the values c_1 and c_2 become closer to each other and the radius Δ grows, then the value of S grows.

In the experiments, we use the same set of the pattern images for the evaluation of S as we have used previously, but the images are presented now in a grayscale mode with the values $c_1 = 110$ and $c_2 = 120$. Table 3 shows the values of S we obtained for various values of Δ. Table 4 shows a growth of S for $c_1 = 119$ and $c_2 = 120$. The percentage S has a tendency to grow as the radius Δ grows, if only the value of $|c_1 - c_2|$ is rather small comparing to Δ.

Note that the algorithm we have presented is not applicable for recognition of raster images subjected to spatial displacements, i.e., rotations or shifts.

In the third computational experiment, we compare the recognition efficiency of heuristics (2) and (7) for recognition problems that we construct in the following way.

Let A be some numerical matrix. Let us construct two matrices, both of which are obtained from A. We shall consider these matrices as pattern matrices denoted by $\mathsf{A}^{(1)}$

Table 2: The results of the recognition tests for the noise level $Q = 45\%$, $n = 50$. The ij-th entry of the table is equal to the recognition efficiency for the experiment with the ordered pair $(\mathsf{A}^{(i)}, \mathsf{A}^{(j)})$, where $\mathsf{A}^{(i)}$ is the numerical matrix that corresponds to the digit i $(i, j = \overline{0, 9})$.

$i \setminus j$	0	1	2	3	4	5	6	7	8	9
0	—	97	92	100	91	86	96	94	85	97
1	98	—	98	98	98	98	94	92	98	99
2	94	94	—	94	99	97	100	96	98	89
3	99	96	96	—	97	96	97	91	100	95
4	100	93	99	96	—	92	99	99	93	95
5	94	97	95	97	93	—	94	99	90	96
6	96	93	100	95	97	94	—	95	99	98
7	99	99	99	91	98	96	96	—	99	99
8	76	92	93	96	89	84	92	97	—	94
9	96	97	93	89	94	95	98	97	96	–

Table 3: The percentage S for the noise level $Q = 44\%$, $c_1 = 110$, $c_2 = 120$.

Δ	10	25	50	75	100
$S, \%$	0	5.4	7.4	16.2	23.5
$P, \%$	100	99.93	99.79	99.72	99.81

and $\mathsf{A}^{(2)}$. As we derive $\mathsf{A}^{(1)}$ from A, we change a rather small number of the matrix A elements. but the changes are large. As we derive $\mathsf{A}^{(2)}$, we change a majority of the matrix A elements, but the changes are relatively small.

Denote by MP the percentage of elements of A that we change obtaining $\mathsf{A}^{(2)}$. $100 - MP$ is the percentage of elements of A that we change obtaining $\mathsf{A}^{(1)}$. Let $\mathsf{A}^{(1)} := \mathsf{A}$, and $\mathsf{A}^{(2)} := \mathsf{A}$. Construct the matrices $\mathsf{A}_M^{(1)}$ and $\mathsf{A}_M^{(2)}$ using randomness in such a manner that MP percent of their elements are 0's and $100 - MP$ percent of them are 1's. Fill these matrices independently. Then, let η_1 and η_2 be random variables that take the values -1 and 1 with equal probabilities, and let ξ_1 and ξ_2 be independent random variables which have uniform distribution on the interval $[0, 1]$. For matrices $\mathsf{A}^{(1)}$ and $\mathsf{A}^{(2)}$, let Δ_1 and Δ_2, respectively, be some predefined radii of the intervals in which elements of the matrices are changing. Let $\Delta_1 > \Delta_2$.

Using elements of $\mathsf{A}_M^{(1)}$, we change $100 - MP$ percent of the elements of $\mathsf{A}^{(1)}$ in accordance with the elements $(\mathsf{A}_M^{(1)})_{ij}$. If $(\mathsf{A}_M^{(1)})_{ij} = 0$, then we do not change the value of $\mathsf{a}_{ij}^{(1)}$. Otherwise,

$$\mathsf{a}_{ij}^{(1)} := \mathsf{a}_{ij}^{(1)} + \lfloor \eta_1 \xi_1 \Delta_1 \rfloor ,$$

where $\lfloor \cdot \rfloor$ denotes the operation of taking an integer part of a real number. Also, let us change MP percent of the elements of $\mathsf{A}^{(2)}$. The positions of these elements are

Table 4: The percentage S for the noise level $Q = 44\%$, $c_1 = 119$, $c_2 = 120$.

Δ	10	25	50	75	100
$S, \%$	22.8	37.5	47.3	46.4	46.4
$P, \%$	99.71	99.6	99.8	99.72	99.82

specified by elements of $(\mathsf{A}_M^{(2)})_{ij}$. If $(\mathsf{A}_M^{(2)})_{ij} = 0$, then

$$\mathsf{a}_{ij}^{(2)} := \mathsf{a}_{ij}^{(2)} + \lfloor \eta_2 \xi_2 \Delta_2 \rfloor\,;$$

otherwise, the value of $\mathsf{a}_{ij}^{(2)}$ stays the same.

Assuming now that the matrix A is the matrix to be recognized and that $\mathsf{A}^{(1)}$ and $\mathsf{A}^{(2)}$ are the pattern matrices. For large values of MP, since the majority of the elements of the matrix A are equal to the corresponding elements of $\mathsf{A}^{(1)}$, it is natural to suggest that the correct recognition is the recognition that gives $\mathsf{A}^{(1)}$ as a result. We can construct such matrices A, $\mathsf{A}^{(1)}$, and $\mathsf{A}^{(2)}$ that

$$\rho_1(\mathsf{A}, \mathsf{A}^{(1)}) > \rho_1(\mathsf{A}, \mathsf{A}^{(2)}), \tag{9}$$

when

$$\delta_{\Xi}(\mathsf{A}, \mathsf{A}^{(1)}) < \delta_{\Xi}(\mathsf{A}, \mathsf{A}^{(2)}). \tag{10}$$

The following matrices give us such an example:

$$\mathsf{A} = \begin{pmatrix} 0 & 0 & 4 & 1 \\ 1 & 3 & 1 & 0 \\ 1 & 2 & 0 & 2 \\ 0 & 4 & 0 & 1 \end{pmatrix}, \quad \mathsf{A}^{(1)} = \begin{pmatrix} 0 & 0 & 4 & 57 \\ 1 & 3 & 1 & 0 \\ 1 & 2 & 0 & 2 \\ 0 & 4 & 0 & 47 \end{pmatrix}, \quad \mathsf{A}^{(2)} = \begin{pmatrix} -3 & 0 & 14 & 1 \\ 5 & -3 & -3 & 8 \\ 8 & 0 & 6 & -1 \\ -2 & -6 & 0 & -7 \end{pmatrix}.$$

Here the values of elements of A are randomly and uniformly chosen from the interval $[0, 5]$, $MP = 90$, $\Delta_1 = 60$, $\Delta_2 = 10$, $\upsilon = 10$. For these matrices, we have

$$\rho_1(\mathsf{A}, \mathsf{A}^{(1)}) = 102 > 73 = \rho_1(\mathsf{A}, \mathsf{A}^{(2)}),$$

while

$$\delta_{\Xi}(\mathsf{A}, \mathsf{A}^{(1)}) \approx 0.312 < 2.359 \approx \delta_{\Xi}(\mathsf{A}, \mathsf{A}^{(2)}).$$

For an experiment, let A be a 10×10-matrix with elements that are randomly and uniformly chosen integers from the interval $[110, 120]$. Taking $MP = 90\%$ and taking the same as above values of Δ_1 and Δ_2, we generate matrices $\mathsf{A}^{(1)}$ and $\mathsf{A}^{(2)}$ such that (9) holds. For more than 95% of the recognition trials, we have (10), and so we have $\mathsf{A}^{(1)}$ as a result of recognition. It is appropriate to accept these results as correct since 90% of corresponding elements of A and $\mathsf{A}^{(1)}$ are equal to each other, while 90% of the corresponding elements of A and $\mathsf{A}^{(2)}$ differ.

As the MP percentage decreases, the results of the recognition become worse for the same values of Δ_1 and Δ_2. The results for some other values of MP and Δ_1 are shown in Table 5. These results demonstrate that, using the developed heuristic, we take into account aggregate variation of the matrix elements rather than large variations of a small number of the elements.

Table 5: The recognition efficiency P for various MP and Δ_2 for the experiment with such matrices A, $\mathsf{A}^{(1)}$, and $\mathsf{A}^{(2)}$ that $\rho_1(\mathsf{A}, \mathsf{A}^{(1)}) > \rho_1(\mathsf{A}, \mathsf{A}^{(2)})$ ($\Delta_1 = 10$ for all the instances).

Δ_2	$MP, \%$	Δ_1	$P, \%$	$MP, \%$	Δ_1	$P, \%$
10	85	35	88	80	25	76
10	85	40	79.8	80	30	67
10	85	45	71.8	80	35	52.8
Δ_2	$MP, \%$	Δ_1	$P, \%$	$MP, \%$	Δ_1	$P, \%$
10	75	20	63	70	15	53
10	75	25	47.1	70	20	36
10	75	30	29.4	70	25	16

5 Conclusions

We have presented an interval approach to pattern recognition of numerical matrices. The heuristic used in the recognition is a minimization of a measure of closeness between two matrices. For computing the measure, we construct an interval linear system associated with the matrices. We take the Lebesgue measure of its solution set as the measure of closeness. Using the heuristic, we construct a recognition algorithm. The recognition algorithm has the best possible quadratic computational complexity if we take interval Gauss-Seidel iteration for computing enclosures of the solution sets.

The computational experiments show that our heuristic produces good results for recognition of raster images and that its recognition efficiency grows as the resolution of the images grows. The experiments also demonstrate that the heuristic takes into account some aggregate variation of matrix elements rather than large variations of a small number of the elements.

Acknowledgements

The author is grateful to Sergey P. Shary for his attention and comments.

References

[1] Oettli, W. On the solution set of a linear system with inaccurate coefficients, *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis.* 1965. Vol. 2, No. 1. P. 115–118.

[2] Shary, S.P. Finite-dimensional interval analysis. (In Russian) URL: `http://www.nsc.ru/interval/Library/InteBooks/SharyBook.pdf`. Accessed 18.11.2013.

[3] Neumaier, A. *Interval Methods for Systems of Equations.* Cambridge: Cambridge University Press, 1990.

[4] Ahlberg, J.H., Nilson, E.N. Convergence properties of the spline fit. *Journal of the Society for Industrial and Applied Mathematics.* 1963. Vol. 11, No. 1. P. 95–104.

[5] Kirnos, E.A. A comparative analysis of morphological methods of image interpretations: An abstract of dissertation of the candidate of physics and mathematics (the speciality 05.13.18 "mathematical modeling, numerical methods and program complexes"). Moscow, 2004. RGB OD, 61:05-1/118.

[6] Kirnos, E.A., Pytèv, Yu.P., Djukova, E.V. Training the Kora-Type Algorithms. *Pattern Recognition and Image Analysis.* 2002. Vol. 12, No. 1. P. 19–24.

[7] Demuth, H., Beale, M. *Neural Network Toolbox for Use with MATLAB. User's Guide. Version 4.* URL: `http://cs.mipt.ru/docs/comp/eng/develop/software/matlab/nnet/main.pdf`. Accessed 18.11.2013.