Machine Learning as a Preprocessing Phase in Discrete Tomography[⋆]

Mihály Gara, Tamás Sámuel Tasi, and Péter Balázs

Department of Image Processing and Computer Graphics
University of Szeged
Árpád tér 2., H-6720, Szeged, Hungary
{gara,ttasi,pbalazs}@inf.u-szeged.hu

Abstract. In this paper we investigate for two well-known machine learning methods, decision trees and neural networks, how they classify discrete images from their projections. As an example, we present classification results when the task is to guess the number of intensity values of the discrete image. Machine learning can be used in *Discrete Tomography* as a preprocessing step in order to choose the proper reconstruction algorithm or – with the aid of the knowledge acquired – to improve its accuracy. We also show how to design new evolutionary reconstruction methods that can exploit the information gained by machine learning classifiers.

Keywords: Discrete Tomography, Machine Learning, Gray-Level Estimation, Evolutionary Algorithm.

1 Introduction

The aim of *Computerized Tomography* (CT) is to obtain information about the interior of objects without damaging or destroying them. Methods of CT (like filtered backprojection or algebraic reconstruction techniques) often require several hundreds of projections to obtain an accurate reconstruction of the studied object [19,22]. Since the projections are usually produced by X-ray, gamma-ray, or neutron imaging, the acquisition of them can be expensive, time-consuming or can (partially or fully) damage the examined object. Thus, in many applications it is impossible to apply reconstruction methods of CT with good accuracy. In those cases there is still a hope to get a satisfactory reconstruction by using *Discrete Tomography* (DT) [20,21].

In DT we assume that the object to be reconstructed is composed of just a few known materials. With this extra information it is often possible to get accurate

[⋆] This research was supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency, the European Union and the European Regional Development Fund. The work of M. Gara was also is supported by the European Union and co-funded by the European Social Fund under the project number TÁMOP-4.2.2/B-10/1-2010-0012. The work of P. Balázs was also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the Hungarian Scientific Research Fund OTKA PD100950.

reconstructions even from a small number of projections. The most important applications of DT arise from the field of industrial non-destructive testing [12] and electronmicroscopy [9]. Very recently, with a technique of DT, the authors of [29] were also able to reveal the 3D structure of crystalline nanoparticles on the atomic scale, which can yield a dramatic breakthrough in nanosciences.

Unfortunately, the DT reconstruction task is usually undetermined, i.e., there can be many different solutions of the same reconstruction task. In order to reduce the number of possible solutions it is often assumed that the image to be reconstructed satisfies some additional (geometrical or more complex structural) properties. There are lots of reconstruction algorithms in DT working in different classes of discrete images defined by certain geometrical or topological properties. For example various kinds of convexity and connectedness are examined in [3,6,7,13,14,16]. However, only few efforts have been made to study how these features can be extracted before the reconstruction, if they are not explicitly given [4,5,17].

Artificial Intelligence (AI) has an extremely broad range of tools for data mining. Surprisingly, up to now, in discrete tomography only a few of them have been used. In most of the cases methods of AI were only used in the reconstruction process itself and not in the preprocessing. For example, in [10] *Neural Networks* and in [8,28] *Genetic Algorithms* were successfully applied for computing reconstructions.

In this paper we study the possibility of retrieving some properties of discrete images from the projections themselves. We investigate for decision trees and neural networks how they perform in classifying discrete images with different structural properties by using the projection data. We also show how to design reconstruction algorithms which can exploit the (often uncertain) knowledge gained by machine learning techniques.

This work summarizes and extends the results of [5] and [17] with the aim of giving an up-to-date insight into the field of applying machine learning in discrete tomography. The structure of the paper is the following. Section 2 is for the problem description of discrete tomography and to introduce the applied learning methods. Section 3 gives an overview of applying machine learning methods in binary tomography. In Section 4 we investigate an important problem of discrete tomography, namely, the identification of the number of gray-intensity values that can be present in the image. In Section 5 we show how the (often uncertain) information gained by machine learning methods can be incorporated into the reconstruction task. Section 6 gives – as a case study – the details of an object-based evolutionary algorithm to solve the reconstruction. Finally, Section 7 is for the conclusion.

2 Preliminaries

2.1 Discrete Tomography

The reconstruction of 3D objects is usually done slice-by-slice, i.e, by integrating together the reconstructions of 2D slices of the object. Such a 2D slice can be represented by a function $f : \mathbb{R}^2 \to \mathbb{R}$. The *Radon transformation* $\mathcal{R}f$ of f is then defined by

$$[\mathcal{R}f](s,\vartheta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\delta(x\cos\vartheta + y\sin\vartheta - s)dxdy \ , \tag{1}$$

where $\delta(\cdot)$ denotes the Dirac delta function, s is the perpendicular distance of a line to the origin, and ϑ is the angle formed by the distance vector. For a fixed angle ϑ we call $\mathcal{R}f_\vartheta(s) : \mathbb{R} \to \mathbb{R}$ as the *projection* of f defined by the angle ϑ. Especially, the projections defined by the angle $\vartheta = 90°$, $\vartheta = 0°$, $\vartheta = 45°$, and $\vartheta = 135°$ are called the *horizontal, vertical, diagonal*, and *antidiagonal projections*, respectively (see Fig. 1 for an example of the horizontal and vertical projections).

The reconstruction problem can be stated mathematically as follows. Given the functions $g(s,\vartheta_1),\ldots,g(s,\vartheta_n)$ (where n is a positive integer) find a function f such that

$$[\mathcal{R}f](s,\vartheta_i) = g(s,\vartheta_i) \quad (i = 1,\ldots,n) \ . \tag{2}$$

In the followings we always assume that the projections are given by a finite sampling, i.e., for each projection direction the line integrals are given along parallel lines (called *projection rays*) with a unit distance between them. In that way each projection can be represented by a vector of $\mathbb{R}$ where m is the number of projection rays in a certain direction.

In discrete tomography we make the assumption that the range of f is a finite set of known discrete values, i.e., $f : \mathbb{R}^2 \to S$ where S is the finite set of values the function f can take. In that case, the image represented by f is called a *discrete image*. In the special case when $S = \{0,1\}$ the task is to reconstruct a binary image, and the field is called *Binary Tomography*.

A discrete image is commonly represented in two ways. In the widely used *pixel-based model*, the image consists of pixels and each pixel can take a value of S. In the *object-based representation* it is supposed that the image contains objects which can be described by their parameters. For example, if the image represents disks, then each disk can be identified by its radius, its center point coordinates, and its gray-intensity value. Such an image can be seen in Fig. 1. The pixel-based model is more general, but it cannot exploit the structural properties of the image.

2.2 Machine Learning Methods

Several machine learning approaches can be used to effectively extract important features of a given object, or even to perform classification tasks to distinguish objects of different type with the aid of the extracted features. In the followings we shortly recall decision trees and neural networks, probably the two most widely used machine learning methods.

Decision trees [27] are special trees which have certain conditional expressions attached to their internal nodes and labels attached to their leaves. These type of trees are primarily used for classification tasks. The way the tree determines the

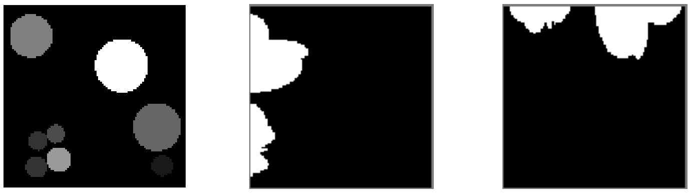


Fig. 1. A discrete image and its horizontal and vertical projections (from left to right, respectively)

class label for an input sample is the following: starting from the root, evaluate the expression attached to the node and move downward to the child node which corresponds to the obtained result. This process is repeated until the reached node is a leaf. In the latter case the input instance gets the label attached to the leaf and the procedure stops. To construct a decision tree a labeled training dataset is needed. Each training sample in this dataset is used to build the previously mentioned expressions assigned to each non-leaf node by determining conditions (most usually, threshold values) for the most significant attributes which separate the input samples the best.

Feed-forward neural networks [26] are widely used classification tools as well. A network consists of several inter-connected units called *neurons*, which are organized in separate layers. Each neuron in the input layer receives a single attribute of the input instance and applies the *activation function* on this value. The produced output for each unit travels along a weighted directed link towards all neurons in the next layer. Weights may differ as they determine the strength and sign of a given connection. In such multilayer configurations every non-input or non-output layer is said to be *hidden*. Each hidden unit sums its inputs and sends the resulting activation value to the next layer as well. Output neurons are exceptional, as they provide the classification result for the given instance.

Training the network consists of numerous *epochs*, where each epoch means a run-through all the samples in the training dataset. During this step the desired output of each training instance is compared to the output of the network, and the error of the classification is measured. Based on this error the initially randomly set weights in the network are updated accordingly. The *learning rate* specifies the scale of this modification, while the *momentum* controls the direction and magnitude of this change according to the previous step. The former is usually decreased continuously through the learning stage, starting from a higher value, while the latter is usually constant during the entire training phase. Training can be stopped once a maximum number of epochs is reached, or the desired accuracy is met. In most cases perfect classification cannot be obtained, since a general pattern that fits the desired output does not exist, unless of course *overfitting* on the training data occurs. The accuracy in the latter case will be misleadingly high, while the used learning method will not be able to correctly classify unseen samples. Needless to say that this case is to be avoided when dealing with learning methods.

Instead of providing our own implementation for neural networks, we decided to take advantage of existing solutions. The open-source WEKA framework has been chosen [18]. WEKA offers several machine learning tools implemented in JAVA, encapsulated in a user friendly GUI that enables users to set various parameters. Among those tools is the class named *Multilayer Perceptron.* This is a realization of a multilayer feed-forward network with back-propagation learning, aided by the momentum technique. We describe a few properties of the implementation here in more detail:

- The activation function, by default, for every neuron is the *sigmoid function* $P(t) = 1/(1 + e^{-t})$.
- Throughout our experiments we used networks with a single hidden layer to connect the input and output layers, only changing the number of hidden units in that particular layer when needed.
- The initial weights for all links between neurons in adjacent, separate layers were randomly set to uniformly distributed values between -0.05 and +0.05.

As described above, the training of the network is actually done by modifying the weights of each link between connected neurons, therefore the formulas for this need to be specified. Let W_{ij} denote the weight of the connection from the i-th unit to the j-th unit. After the k-th training sample this weight is updated by $W_{ij} = W_{ij} + \Delta W_{ij}^k$, where

$$\Delta W_{ij}^k = \alpha \cdot Err_{ij}^k + \beta \cdot W_{ij}^{k-1}. \tag{3}$$

In (3) α is the learning rate, Err_{ij}^k is the error measured on the k-th training sample according to the simple backpropagation rule [26], β is the momentum, and W_{ij}^{k-1} is the previous weight change. For the specific values of the required parameters see Table 3 in Section 4.2, where experimental results are presented.

3 Machine Learning Methods in Binary Tomography

The first attempt to apply machine learning methods in discrete tomography was published in [10] where the authors designed neural networks to reconstruct binary images from their projections. It turned out that this approach (although it can produce reconstructions of good quality) has several limitations, and especially even for moderate-sized images the following drawbacks are mentioned:

- the network must usually be huge to ensure accurate reconstructions, i.e., it often has hundreds of inner nodes yielding many connections, too,
- millions of training examples are needed for the learning of the network,
- often 10-20 projections are required for an accurate reconstruction.

As opposed to that work our aim is not to reconstruct the image, but rather to predict its geometrical or other structural properties. Many algorithms are published in the literature of DT which can cleverly exploit such prior information, like e.g., convexity or connectedness of the image (see, e.g., [2] and the references given there). However, all of them make the assumption that this information is explicitly given. *But what can we do if this knowledge is not available before the reconstruction?* Attempting to apply all existing reconstruction algorithms developed for different classes of images, and choose the best reconstruction is obviously not a clever way. Unfortunately, up to now, there are just a few characterization results on how the projections of an image should look like, if the object satisfies some geometrical properties (see, e.g., [15]). But even if there is no chance to give an exact mathematical characterization of an image feature based solely on the projection data, we still have a hope to predict image properties if we use machine learning techniques.

As described before, decision trees and neural networks are effective tools to classify objects of the same type with the aid of their attributes. In discrete tomography the projections can serve as attributes of the discrete images. The question is, whether it is possible to reveal (with an acceptable error) the image properties from those attributes by using the above mentioned learning techniques. If so, then – applying those methods as a preprocessing step – we can improve the speed and quality of the reconstructions.

In our previous work [17] we achieved promising results in revealing the so-called *hv*-convexity property of binary images. In the experiments we used the well-known C4.5 decision tree and a simple feed-forward neural network with one hidden layer and back-propagation learning. Binary images of size $m \times n$ were represented by an $(m + n)$-dimensional feature vector $(h_1, \ldots, h_m, v_1, \ldots, v_n)$ formed by their horizontal and vertical projections. In the classification the feature vectors were used as the input patterns for both studied learning algorithms. It turned out that – with the aid of these learning methods – the *hv*-convex images can be successfully separated from the random ones and from the almost *hv*-convex ones, exclusively based on the projection data. For more details the reader is referred to [17].

4 Determining the Number of Distinct Intensity Levels in Discrete Tomography

Discrete tomography utilizes the strong assumption that the image to be reconstructed contains just a few gray-intensity values that are known beforehand.

Determining the intensity levels is seemingly one of the most difficult problems in discrete tomography. In [12] the authors suggested to reconstruct the discrete image with many intensity levels, and then to perform a second reconstruction with the gray-intensity values defined by the peaks of the histogram of the image obtained in the previous reconstruction. In [11] a semi-automatic method was proposed to select the intensity values. However, up to now, no general method is known to solve this task. In this section we investigate a closely related problem. We study how machine learning can be used for determining the number of intensity values present in the discrete image, at least for a restricted class of images. For decision trees we choose the C4.5 implementation [27], while for the neural networks we use the aforementioned Multilayer Perceptron of the WEKA toolbox [18].

4.1 Generated Datasets

In the experiments we used the horizontal and vertical projections, thus the attributes of each learning instance – for both the decision trees and neural networks – were the coordinates of those two projections. In the following we will call a set of disks with fixed size and position as a *configuration*. That is, instances of the same configuration differ only in the intensity values used in the image. We performed the classification with 100 different configurations for decision trees and 10 different ones for neural networks. Each configuration contained 8 randomly generated disjoint disks with fix positions and equal – at least 5 unit long – radius for that particular configuration (for an example see again Fig. 1). For classification purposes the generated training and testing datasets contained 3600 and 1200 images, respectively, for every configuration.

The reason of the difference in the number of examined configurations between the decision trees and neural networks is the required training time. Training a decision tree is much faster, and as a consequence takes significantly less time than training a neural network, mainly because of additional parameters to set.

Beside the background intensity (that was 0 in every case), futher intensities of the disks were randomly chosen from a given intensity list. Two alternate lists were used for decision trees, one containing equidistant and another one with fixed non-equidistant points in $[0, 1]$ defining the grayscale values. Table 1 shows the intensity lists for a given number of intensity values used in our experiments. For our experiments with neural networks we used equidistant intensities only.

Table 1. The intensity lists for a given number of intensity values (first column) used in our experiments. The background intensity is not counted, and it is always 0.

	equidistant								non-equidistant							
3:	0.1	0.2	0.3						0.1	0.2	0.95					
4:	0.1	0.2	0.3	0.4					0.1	0.2	0.92	0.95				
5:	0.1	0.2	0.3	0.4	0.5				0.1	0.2	0.21	0.92	0.95			
6:	0.1	0.2	0.3	0.4	0.5	0.6			0.1	0.2	0.21	0.9	0.92	0.95		
7:	0.1	0.2	0.3	0.4	0.5	0.6	0.7		0.1	0.2	0.21	0.22	0.9	0.92	0.95	
8:	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.1	0.2	0.21	0.22	0.3	0.9	0.92	0.95

4.2 Experimental Results

In our experiments we used two types of error measurement. The first one is the common, strict method to calculate classification errors: each misclassification is treated as an error. In this case only the diagonal elements of the classification matrix belong to the correctly classified cases (dark gray elements of Table 2). Hereunder we call this method the *normal* error measurement. The second one is a more permissive type of measure. In this case if the difference between the output of the classifier and the exact number of distinct intensities is not greater than 1, the result is accepted. For example for a given image with 4 different intensity values, outputs 3, 4, and 5 are all treated as correct classifications (none of the gray elements of Table 2 are misclassifications).

Table 2. The average of 100 classification matrices of all configurations for decision trees (a), and the average of 10 classification matrices for neural networks (b) for 1-6 equidistant intensity values. The numbers in brackets in the last column represent the exact number of intensities in the image, while in the first row they show the number of intensities estimated by the machine learning. Matrix entries are given in percentage (rounded to two digits) of the test cases for each number of intensities.

(a) Decision tree

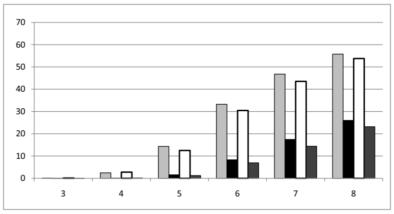
(1)	(2)	(3)	(4)	(5)	(6)	← classified as
100.00	0.00	0.00	0.00	0.00	0.00	(1)
0.31	83.84	11.15	2.94	1.21	0.56	(2)
0.20	14.35	56.25	18.48	7.51	3.23	(3)
0.03	4.69	18.77	46.98	20.6	8.94	(4)
0.00	1.89	7.34	20.77	45.92	24.09	(5)
0.00	0.62	2.79	8.06	21.03	67.51	(6)

(b) Neural network

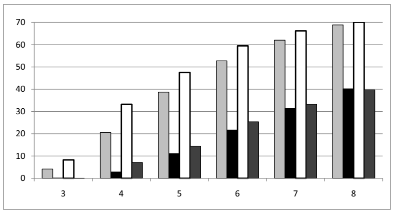
(1)	(2)	(3)	(4)	(5)	(6)	← classified as
100.00	0.00	0.00	0.00	0.00	0.00	(1)
0.00	98.90	1.00	0.10	0.00	0.00	(2)
0.20	2.30	91.85	4.20	0.85	0.60	(3)
0.00	0.75	3.10	75.90	12.35	7.90	(4)
0.00	0.00	0.70	3.90	95.40	0.00	(5)
0.00	0.00	0.00	0.00	0.00	100.00	(6)

Table 2 shows the average of the 100 acquired classification matrices for decision trees, and the average of 10 classification matrices for neural networks on classifying images with 1-6 distinct intensity levels. The dark gray elements represent the good classifications in normal measurement, while during the permissive measurement mode every case that corresponds to a gray-shaded element in a row is accepted as a correct classification.

We also investigated the robustness of the presented methods by performing the same experiments but this time with noisy projection data. In these tests we used additive noise with uniform distribution with a noise ratio of 5%. Comprehensive results are shown in Figs. 2 and 3 for decision trees and neural networks, respectively. With decision trees we attempted to distinguish 3 to 8

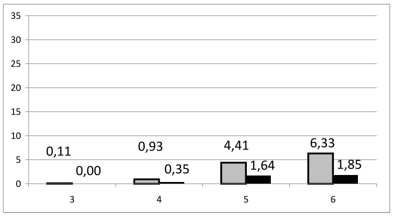


(a) Noiseless

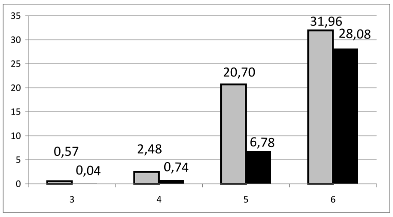


(b) With 5% noise

Fig. 2. Classification error of the decision trees depending on the number of different intensity levels in the image without (a) and with (b) noise. For each number of intensity the bars represent the average error for the equidistant intensity values with normal (light gray) and permissive (black) error measurement, and for the non-equidistant values with normal (white) and permissive (dark gray) error measurement.



(a) Noiseless



(b) With 5% noise

Fig. 3. Classification error of the neural networks depending on the number of different intensity levels in the image without (a) and with (b) noise. For each number of intensities the plotted bars represent the average error for equidistant intensity values with normal (gray) and permissive (black) error measurement.

intensity levels at once, while for neural networks we only examined 3 to 6 different levels (as seen in Fig. 3). The main reason behind the reduction in the latter case was the extremely long training time. While the building of decision trees is relatively fast, finding the proper weights of neural networks is a much slower task. Nevertheless, practical applications of discrete tomography usually involve no more than 4 or 5 intensity levels. On the other hand, neural networks have their advantages as well. The most important one is that we gained much better classifications with neural networks than with decision trees for the investigated problem. This is probably due to the various parameters available to configure the method in an optimal way. Careful adjustments of these variables – such as learning rate, momentum, etc. – can lead to better classification results. To find the parameters close to the best possible we tested several settings for each dataset. By modifying one parameter at once, and observing its effect on the classification result on the training data, we tried to keep track of each parameters optimal direction of change. This way in some cases a clear pattern has been found how to set up the network properly. The average of used parameter setups are displayed in Table 3. α is usually decreased as the training goes on, and this learning rate decay in this implementation is achieved by dividing the learning rate after each epoch by the number of epochs completed so far. Thus, Table 3 contains the averages of the initially set learning rates. The momentum, on the other hand, did not change during training.

Table 3. Average values of the parameters of the neural network classification

Noiseless

#intensities	Learning rate	Momentum	Training time	Hidden neurons
3	0.2	0.8	100	10.5
4	0.24	0.78	190	16
5	0.27	0.75	370	41
6	0.238	0.8275	530	55.5

5% Noise

#intensities	Learning rate	Momentum	Training time	Hidden neurons
3	0.2	0.8	100	10
4	0.3	0.8	200	20
5	0.27	0.75	740	41
6	0.2218	0.8275	133	54

We drew the conclusion from our experiments on neural networks that we had to increase the number of hidden neurons as the number of intensities increased. In the noiseless case increasing the training time (number of epochs) provided better results. On the noisy dataset longer training time gave worse results, probably because the network tended to overfit the given training samples, therefore it was not be able to generalize as well as expected.

Our experiments on decision trees contained also datasets with non-equidistant intensity values. These tests provided almost the same results as the equidistant ones. However, the charts of Fig. 2 reveal that – as one could expect – the non-equidistant scenarios were more sensitive to noise.

5 Discrete Reconstruction with Learnt Priors

The reconstruction task can be reformulated as an optimization problem where the aim is minimize

$$\Phi(\mathbf{x}) = \lambda_1 ||\mathbf{A}\mathbf{x} - \mathbf{b}|| + \lambda_2 \varphi(\mathbf{x}) \ , \quad (4)$$

where $\mathbf{A}$ is the projection geometry matrix which describes the interaction of the projection rays with the image pixels. Here, a_{ij} gives the weight (in our case the length of the line segment) of the i-th projection ray on the j-th image pixel, $\mathbf{b}$ is the vector of all projection values, and $\mathbf{x}$ is the unknown discrete image (given in vector form). The $\varphi(\mathbf{x})$ term stands for the prior (learnt) information. If $\mathbf{x}$ and $\mathbf{y}$ are two images such that $\mathbf{x}$ satisfies the prior information better than $\mathbf{y}$, then $\varphi(\mathbf{x}) < \varphi(\mathbf{y})$. Finally, $\lambda_1, \lambda_2 \geq 0$ are suitably chosen weighting factors, to control the confidence of the data fidelity and the learnt information, respectively. That is, if the classification of a machine learning method shows a high degree of uncertainty, then a smaller λ_2 value is reasonable, while for more trusted classifications λ_2 can be set higher.

Unfortunately, the discrete optimization problem is in general NP-hard, thus (4) is usually solved by approximation techniques and/or heuristics, like e.g. simulated annealing [25] or evolutionary algorithms [1]. In the following section we give a case study of optimizing (4) by an evolutionary approach.

6 Optimization with an Object-Based Evolutionary Algorithm: A Case Study

6.1 The Evolutionary Algorithm

Equation (4) can be minimized in numerous ways. In [5] we developed an object-based evolutionary algorithm to reconstruct binary images containing disks inside a ring from their horizontal, vertical, diagonal and antidiagonal projections. Figure 4 shows such an image with its projections. These images are typical in analyzing reconstruction algorithms for non-destructive testing [23].

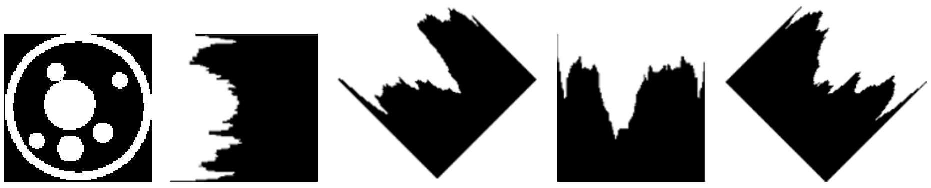


Fig. 4. An example of our test images and its horizontal, diagonal, vertical, and antidiagonal projections (left to right, respectively)

Our method deals with a set of image entities which are represented not by an array of pixels but rather by the coordinates and radius of the circles. There are two ways the entities can be modified, the *mutation* and the *crossover*. During mutation a disk may be removed or a new one can be added to the image, and the radius or the center point coordinates of a disk can be modified. The crossover operation mixes the disks of two image entities. After mutation and crossover, in every generation there is a selection period, in which the elements are sorted by their fitness values and only the given number of them stay alive in the next generation. The fitness value is based on the form what we would like to minimize. In our method it is simply calculated by the formula of (4), thus smaller fitness values belong to better solutions.

6.2 Incorporating the Learnt Knowledge

Again, we used the C4.5 decision tree to estimate the number of disks in an image, in advance. As we knew that each test image contained 1-10 disks we could suit the attributes of the learning method to this information. Of course such settings are generally not possible. Nevertheless, it also shows the flexibility of the approach. We tried many different ways to define the attributes for the C4.5 classifier. We found that the best results were provided by taking the local maxima of the projections. For every projection we identified the number of local maxima and – in addition – we also took the first 10 maximum values as attributes (if they existed). In this way – as Table 4 shows – the classification error was relatively high for certain number of disks if only the real value was considered as a correct classification. However, if we accepted a difference of 1 or 2 between the real and the estimated value, we got quite good classifications (see again Table 4).

Table 4. Estimating the number of disks. Error of classification in percentage if no difference (second row) or a difference of 1 (third row) or 2 (fourth row) is allowed between the real and the estimated value. The first row represents the real number of disks present in the image.

	1	2	3	4	5	6	7	8	9	10
0	0	8	25	51	79	65	74	70	75	56
1	0	0	1	5	32	19	27	31	20	27
2	0	0	0	0	5	7	5	3	0	0

We incorporated the learnt prior information into the optimization task of (4) in the form

$$\varphi(\mathbf{x}) = 1 - \frac{t_{c_{\mathbf{x}},c}}{\sum_{i=1}^{10} t_{i,c}} , \tag{5}$$

where $c_{\mathbf{x}}$ was the number of disks in the image $\mathbf{x}$, c was the expected number of disks given by the decision tree, and t_{ij} denoted the number of test examples which had i disks but were classified as containing j of them. For more details on the classification errors and the description of the algorithm (together with its parameters) see [5].

6.3 Parameter Settings

This section focuses on settig certain general parameters of the evolutionary algorithm, not studied in details in [5]. For finding a more robust configuration independent from any learning result, in these experiments we omitted the learnt priors by setting $\lambda_1 = 1$ and $\lambda_2 = 0$. We only used two restrictions for the number of disks: there were at least 1 and at most 12 disks in the image. The algorithm stopped when the number of generations (iterations) reached its maximum value – 250, 500, 750, and 1000.

In the experiments the size of the initial population was 250, 500, 750, and 1000. We worked with constant population size, i.e. in each generation we kept a fixed number of the best entities (equal to the size of the initial population). The ranking of the instances were based on their fitness values. For creating the initial population we used the generation algorithm from the DIRECT system [24]. The initial populations contained elements from each class in the same ratio. For example, if the size of the population was 250 then it contained 25-25 entities with 1, ..., 10 disks.

The probability value for increasing or decreasing the number of disks was 0.05, for both changing their radius or position it was 0.5-0.5, and for the crossover it was, again, 0.05. In the model only disjoint disks (and a disjoint ring) were allowed. If the operators resulted in intersecting ones we dropped the image and tried to generate a new one. The number of attempts was at most 50 in crossover phase and 1000 in case of adding a new disk to the image.

6.4 Measuring the Quality of the Reconstructions

To measure the error of the reconstruction there is a widely used error rate, the Relative Mean Error (RME) that is given by

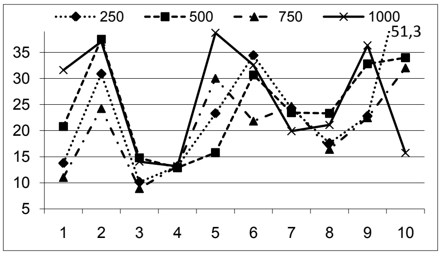
$$RME = \frac{\sum_i |f_i^o - f_i^r|}{\sum_i f_i^o} \cdot 100\% \ , \tag{6}$$

where f_i^o and f_i^r denote the i-th pixel of the original and the reconstructed image, respectively. Thus the RME value gives the error in proportion to the white pixels on the original image. Lower RME value means better reconstruction. Note that – especially if there are just a few object points in the original image – RME can also be greater than 100%. This yields relatively higher RME values of images with relatively few white (object) points.

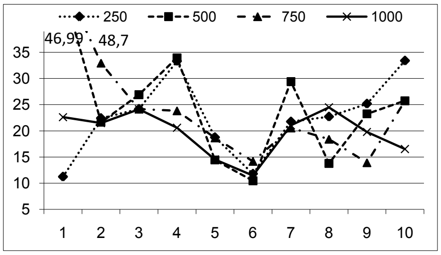
Since our reconstruction model is object-based, to calculate (4) and to use the abovementioned error metric we have to discretize the image. We also investigated the correlation of the measured error and the image resolution and we found that – above a reasonable level which is, say 200×200 – the resolution had no significant effect on the reconstructions, and their RME values.

6.5 Experimental Results of Parameter Settings

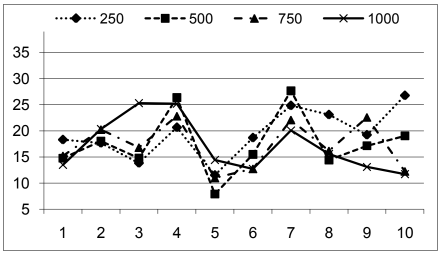
The reconstruction errors measured in our experiments are shown in Fig. 5. On the charts we indicated the average errors depending on the number of disks in the image for the different population sizes, which were respectively 250 (a), 500 (b), 750 (c) and 1000 (d). Each value is calculated as the average of



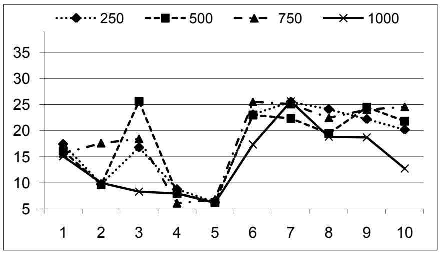
(a)



(b)



(c)



(d)

Fig. 5. Reconstruction error for different population sizes and generation numbers. The population sizes were 250 (a), 500 (b), 750 (c) and 1000 (d). The curves on the charts show the RME values (vertical axis) for the different generation numbers depending on the number of disks in the image (horizontal axis). The test dataset contained 5-5 images with 1, 2, . . ., 10 disks.

5 reconstructions, performed on 5 different images with the same number of disks. The different number of generations provide similar results for a given population size, but comparing the individual charts we can deduce that the reconstruction quality is more dependent on the population size, and of course the bigger the population the lower the resulting RME values become.

The difference between the reconstructed and the original number of disks depending on the population size and the number of generations is presented in Table 5(a). We can observe that the number of cases where the reconstructed image contained fewer or more disks than the original on decrased as the number of generations (see rows of Table 5(a)) and the size of the population (columns of Table 5(a)) grew. Table 5(b) provides information about the reconstruction time. It can be seen that by increasing the number of generations and/or the population size the running time increases, too, which is in accordance of the intuition and a straight consequence of the general design of all evolutionary algorithms.

Table 5. (a) Percentage of the test cases where the number of disks differ between the reconstructed and the original image, for different population sizes (columns) and generation numbers (rows). (b) The time of reconstruction in seconds for different population sizes (columns) and generation numbers (rows). The test image contained 10 disks.

(a)

	250	500	750	1000
250	44	38	24	24
500	32	34	16	31
750	26	22	10	12
1000	28	16	12	10

(b)

	250	500	750	1000
250	2	3	5	7
500	3	6	12	22
750	5	17	27	36
1000	10	37	37	47

7 Conclusion

With the aid of machine learning various properties of discrete (binary) images can be extracted from their projections. We applied decision trees and neural networks for this task. As an example we investigated here the problem of estimating the number of intensity values of an image, solely based on the projection data. The information gained by learning techniques can be exploited in discrete tomography by designing methods capable of incorporating the (often uncertain) learnt information into the reconstruction process. One way to do this is to reformulate the reconstruction problem to an optimization task and to solve it by evolutionary algorithms. In this paper we gave an up-to-date overview of our work on this topic. Our results seem to be promising and they can hopefully be useful in solving even more difficult problems of discrete tomography, like e.g. the estimation of gray intensities in a discrete image, which nowadays is one of the most challenging problems in the field of discrete image reconstruction.

References

1. Bäck, T., Fogel, D.B., Michalewicz, T. (eds.): Evolutionary Computation 1. Institute of Physics Publishing, Bristol and Philadelphia (2000)
2. Balázs, P.: Binary Tomography Using Geometrical Priors: Uniqueness and Reconstruction Results. PhD thesis at the University of Szeged, Szeged (2007), `http://www.inf.u-szeged.hu/~pbalazs/research/bp_thesis_main.pdf`
3. Balázs, P., Balogh, E., Kuba, A.: Reconstruction of 8-connected but not 4-connected hv-convex discrete sets. Disc. Appl. Math. 147, 149–168 (2005)
4. Balázs, P., Gara, M.: Decision Trees in Binary Tomography for Supporting the Reconstruction of *hv*-Convex Connected Images. In: Blanc-Talon, J., Bourennane, S., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2008. LNCS, vol. 5259, pp. 433–443. Springer, Heidelberg (2008)
5. Balázs, P., Gara, M.: An Evolutionary Approach for Object-Based Image Reconstruction Using Learnt Priors. In: Salberg, A.-B., Hardeberg, J.Y., Jenssen, R. (eds.) SCIA 2009. LNCS, vol. 5575, pp. 520–529. Springer, Heidelberg (2009)
6. Balogh, E., Kuba, A., Dévényi, C., Del Lungo, A.: Comparison of algorithms for reconstructing hv-convex discrete sets. Lin. Algebra and its Applications 339, 23–35 (2001)

7. Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Medians of polyominoes: A property for the reconstruction. Int. J. Imaging Systems and Techn. 9, 69–77 (1998)
8. Batenburg, K.J.: An evolutionary algorithm for discrete tomography. Discrete Appl. Math. 151, 36–54 (2005)
9. Batenburg, K.J., Bals, S., Sijbers, J., Kuebel, C., Midgley, P.A., Hernandez, J.C., Kaiser, U., Encina, E.R., Coronado, E.A., Van Tendeloo, G.: 3D imaging of nanomaterials by discrete tomography. Ultramicroscopy 109(6), 730–740 (2009)
10. Batenburg, K.J., Kosters, W.A.: A Neural Network Approach to Real-Time Discrete Tomography. In: Reulke, R., Eckardt, U., Flach, B., Knauer, U., Polthier, K. (eds.) IWCIA 2006. LNCS, vol. 4040, pp. 389–403. Springer, Heidelberg (2006)
11. Batenburg, K.J., Van Aarle, W., Sijbers, J.: A semi-automatic algorithm for grey level estimation in tomography. Pattern Recognition Letters 32, 1395–1405 (2011)
12. Baumann, J., Kiss, Z., Krimmel, S., Kuba, A., Nagy, A., Rodek, L., Schillinger, B., Stephan, J.: Discrete tomography methods for nondestructive testing. In: [21], pp. 303–331 (2007)
13. Brunetti, S., Daurat, A.: An algorithm reconstructing convex lattice sets. Theor. Comput. Sci. 304, 35–57 (2003)
14. Brunetti, S., Del Lungo, A., Del Ristoro, F., Kuba, A., Nivat, M.: Reconstruction of 4- and 8-connected convex discrete sets from row and column projections. Lin. Alg. Appl. 339, 37–57 (2001)
15. Castiglione, G., Frosini, A., Restivo, A., Rinaldi, S.: A Tomographical Characterization of L-Convex Polyominoes. In: Andrès, É., Damiand, G., Lienhardt, P. (eds.) DGCI 2005. LNCS, vol. 3429, pp. 115–125. Springer, Heidelberg (2005)
16. Chrobak, M., Dürr, C.: Reconstructing hv-convex polyominoes from orthogonal projections. Inform. Process. Lett. 69(6), 283–289 (1999)
17. Gara, M., Tasi, T.S., Balázs, P.: Learning connectedness and convexity of binary images from their projections. Pure Math. and Appl. 20, 27–48 (2009)
18. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1), 10–18 (2009)
19. Herman, G.T.: Fundamentals of Computerized Tomography: Image reconstruction from projections. Springer, Heidelberg (2009)
20. Herman, G.T., Kuba, A. (eds.): Discrete Tomography: Foundations, Algorithms and Applications. Birkhäuser, Boston (1999)
21. Herman, G.T., Kuba, A. (eds.): Advances in Discrete Tomography and its Applications. Birkhäuser, Boston (2007)
22. Kak, A.C., Slaney, M.: Principles of Computerized Tomographic Imaging. IEEE Press, New York (1999)
23. Kiss, Z., Rodek, L., Kuba, A.: Image reconstruction and correction methods in neutron and X-ray tomography. Acta Cybernetica 17(3), 557–587 (2006)
24. Kuba, A., Ruskó, L., Kiss, Z., Nagy, A.: Discrete Reconstruction Techniques. Electronic Notes in Discrete Math. 20, 385–398 (2005)
25. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, E.: Equation of state calculation by fast computing machines. J. Chem. Phys. 21, 1087–1092 (1953)
26. Mitchell, T.M.: Machine Learning. McGraw Hill, New York (1997)
27. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
28. Valenti, C.: A genetic algorithm for discrete tomography reconstruction. Genet. Program Evolvable Mach. 9, 85–96 (2008)
29. Van Aert, S., Batenburg, K.J., Rossell, M.D., Erni, R., Van Tendeloo, G.: Three-dimensional atomic imaging of crystalline nanoparticles. Nature 470, 374–377 (2011)