INFORMATICA, 2007, Vol. 18, No. 3, 343–362
© 2007 *Institute of Mathematics and Informatics, Vilnius*

# Assessment of Classification Models with Small Amounts of Data

Boštjan BRUMEN, Matjaž B. JURIČ, Tatjana WELZER, Ivan ROZMAN
*Faculty of Electrical Engineering and Computer Science, University of Maribor*
*Smetanova 17, Si-2000 Maribor, Slovenia*
*e-mail: bostjan.brumen@uni-mb.si*

Hannu JAAKKOLA
*Tampere University of Technology, Pori*
*PO BOX 300, Fi-28101 Pori, Finland*

Apostolos PAPADOPOULOS
*Department of Informatics, Aristotle University*
*PO BOX 451, Thessaloniki, GR-54124, Greece*

Received: October 2006

**Abstract.** One of the tasks of data mining is classification, which provides a mapping from attributes (observations) to pre-specified classes. Classification models are built by using underlying data. In principle, the models built with more data yield better results. However, the relationship between the available data and the performance is not well understood, except that the accuracy of a classification model has diminishing improvements as a function of data size. In this paper, we present an approach for an early assessment of the extracted knowledge (classification models) in the terms of performance (accuracy), based on the amount of data used. The assessment is based on the observation of the performance on smaller sample sizes. The solution is formally defined and used in an experiment. In experiments we show the correctness and utility of the approach.

**Key words:** assessment, classification, accuracy, learning curve, sampling.

## 1. Introduction

The ability to distinguish between objects is the fundamental to learning and intelligent behavior in general. Being able to act intelligently in an environment relies basically on the ability to distinguish things. If we can distinguish things and events – to classify them according to their specific features, we are able to react. Thus, the study of relations between objects (i.e., their dominant features) is interesting. The similarities and discrepancies are many times the information we seek. They can only be found based on past experiences (knowledge), adequate observing (where the presence of contrast is essential) and the ability to distinguish among objects. We humans are able to find solutions to complex problems based on our own experiences, experiences of other humans and information. The term "knowledge" thus denotes the results of such a further analysis of the information (patterns) (Witten and Frank, 2000). Automated extraction of knowledge has been in interest ever since the advent of computing and has regained the focus of the research community in the last decade with several successes in the area of data mining.

Classification, the most common data mining task, seems to be a human imperative (Berry and Linoff, 1997). In order to understand and to communicate about the world, and to cope with its complexity, we are constantly classifying things. Classification provides a mapping from attributes (observations) to pre-specified groupings or classes. Classification is considered to be supervised learning (Cabena *et al.*, 1997). For supervised learning, the model takes in the independent variables, produces a guess for the dependent variable that is compared to the actual dependent variable and an error-correction is made; hence, the study is supervised. Supervised learning is a process of automatically creating (by using a learning algorithm) a model from a set of records (examples) called a training set. In other words, it is a process of extracting the knowledge from data.

The performance of the model can be measured in the terms of accuracy – the number of correctly classified items over the total number of items in a set. The problem is that there is no way of telling in advance, what the performance will be on the given (real life!) data. Namely, the final performance may be below the requirements of the user, so the early assessment may prevent waste of time and other resources. In this paper, we address the problem of assessing the performance of classification models based on small amounts of data.

*Paper contribution*. In the paper, we give a method for an early assessment of the classification performance. The method is based on the solid formal background, which is extended with the observations of the learning curve while the models are built with smaller sample sizes.

*Paper organization.* In the following section we outline and further define the open research problem. In Section 3, we describe a formal setting for the solution of the problem and in Section 4 we develop a solution based on the formal description. With the solution we are able to make estimates for model's performance – we describe our experiment and the results in Section 5. We conclude the paper with final remarks in Section 6.

## 2. Problem of Learning in Small $l$ Regime

The learning curve depicts a relationship between the sample size and the model performance (see Fig. 1). The horizontal axis represents $l$, the number of instances in a given training set ($l$ can vary between zero and $L$, the total number of available instances). The vertical axis represents the performance (e.g., accuracy, or error rate) of the model produced by an algorithm when given a training set of size $l_i$.

We can model the learning curve with a function, which takes in as an independent variable the sample size $l$. Many authors have tried to do so with a limited set of functions (see, e.g., Harris-Jones and Haines, 1997; Frey and Fisher, 1999). When measuring the accuracy of a classification model, we build a so-called learning curve.

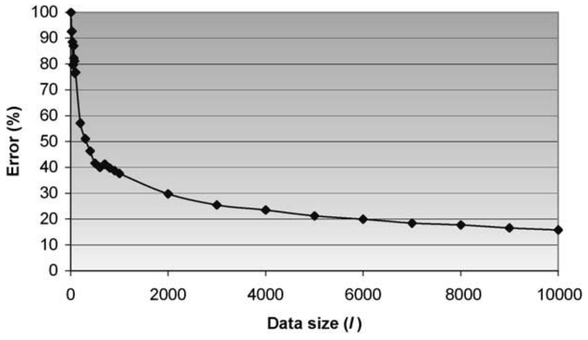

Fig. 1. A typical learning curve.

Learning curves typically have a steeply sloping portion early in the curve, a more gently sloping middle portion, and a plateau late in the curve. This resembles the way humans learn – Anderson and Schooler reviewed a number of paradigms in which a power law appears to describe human performance (Anderson and Schooler, 1991). For this reason the curve is called a learning curve. The plateau level can be considered as the capacity or the final performance of the combination of the data and the learning algorithm (Brumen *et al.*, 2001a; Brumen *et al.*, 2005b).

In order to plot the learning curve we need to measure the performance of a learning algorithm for various sample sizes. The research question here is whether (and how) is it possible to estimate the performance of a learning algorithm (more specifically, the model it builds) without too many measurements. Namely, doing the measurements is costly, especially when the human experts need to prepare the data (Brumen *et al.*, 2002a; Brumen *et al.*, 2003; Brumen *et al.*, 2005a). Thus, we need to do as few measurements as possible to gain an insight into an algorithm's future performance. For this reason we propose an approach where we do the measurements with low amounts of data and try to make an estimate.

In the continuation, we will describe the problem of learning when the amount of data is low. First, we will give a definition of a database, where the data (the training set) – to be fed into a learning algorithm – reside. Next, we formalize the learning algorithm's performance, specifically the error rate. With this set-up, we explain why the work from others can not solve the problem and outline the solution for making estimates when learning with small amounts of data.

DEFINITION 1 (relational database). A relational database $\mathcal{R}$ is a set of relational tables, $\mathcal{R} = \{T_1, \ldots, T_i, \ldots, T_l\}$. Each relational table (also called search table) $T_i$ consists of a table scheme $TS_i$, which is invariant over the life of the table and, at any point of time, a set of tuples consistent with the scheme. Such a set of tuples is called the current value or instance of table $T_i$.

DEFINITION 2 (relational database schema). The *relational database schema* $\mathcal{RS}$ is described by a set of *relational table schemas*, $\mathcal{RS} = \{TS_1, \ldots, TS_i, \ldots, TS_l\}$. A table schema of $T_i, TS_i$, is described by a finite set of *attributes*, $TS_i = \{A_{i1}, \ldots, A_{in}, \ldots, A_{iN}\}$. Each attribute $A_{in}$ is the name of a role played by some domain $D_{in}$ in the relation schema $TS_i$. We assume that any attribute name $A_{in}$ appears only once within table schema $TS_i$. $D_{in}$ is called the domain of $A_{in}$ and is denoted by $D_{in} = \mathrm{dom}\,(A_{in})$. A *domain* is simply a set of values and can be finite or infinite. In a relational database, we have a set of domains, $D = \{D_1, \ldots, D_k, \ldots, D_K\}$. A *table* $T_i$ consists of a table scheme $TS_i$ and a set of $N$-tuples $t_i$, $t_i = \{t_{i1}, \ldots, t_{im}, \ldots, t_{iM}\}$. Each $N$-tuple $t_{im}$ (belonging to the search table $T_i$) is an ordered list of $N$ values, $t_{im} = \langle \nu_{im1}, \ldots, \nu_{imn}, \ldots, \nu_{imN} \rangle$, where each value $v_{imn}$ is an element of $\mathrm{dom}\,(A_{in})$, or is a special *null* value. The $n$th value of the $m$th tuple ($t_{im}$) of search table $T_i$, which corresponds to the attribute $A_{in}$, is referred to as $\nu_{imn} = t_{im}(A_{in})$.

DEFINITION 3 (training set). The *training set* consists of $l$ pairs $z_i \in Z$:

$$Z = \{(x_i, y_i) | x_i \in TS_1 \times TS_2 \times \ldots \times TS_k, 1 \leqslant k \leqslant I \wedge y_i \in \mathbb{R}, 1 \leqslant i \leqslant l\}, \tag{1}$$

where $x_i$ are feature vectors from the database and the labels $y_i$ are continuously valued for regression tasks and discrete (e.g., Boolean) for classification tasks. The training set is assumed to be generated through random independent samples from a stationary but unknown probability distribution:

$$P(x, y) = P(y|x) \cdot P(x). \tag{2}$$

The probability $P(x)$ describes the region of interest in the input space (the database) and the distribution $P(y|x)$ describes the desired functional dependency.

The learning algorithm (machine) is characterized by the family of functions $f_w$ it can realize; the family is parameterized through weights $w \in \mathcal{R}^m$. The goal of the learning is to pick a function from the family $f_w$ it can realize such that the generalization error is minimized, for given size of region of interest $l$:

$$\varepsilon_{gen}(w) = \int \mathrm{d}x\, \mathrm{d}y \cdot P(x, y) \cdot \varepsilon(w, (x, y)), \tag{3}$$

where $\varepsilon(w, (x, y))$ is the error or cost function which measures the loss on pattern $(x, y)$ when the parameters of the learning algorithm are set to $w$. The generalization error measures the expected value of the performance of the implementation $w$ of the learning algorithm.

Since the distribution $P(x, y)$ is unknown, the only way to access $\varepsilon_{gen}(w)$ for a fixed family of functions $f_w$ is through the error on the training set:

$$\varepsilon_{tr}(w, l) = \frac{1}{l} \sum_{i=1}^{l} \varepsilon(w, (x_i, y_i)), \tag{4}$$

where $l$ is the size of the training set.

The law of large numbers ensures that the difference between the mean error on the training set and the generalization error of the learning algorithm goes to zero as the size $l$ of the training set goes to infinity for fixed setting of the parameters $w$ (Vapnik, 1982).

$$\lim_{l \to \infty} \big(\varepsilon_{gen}(w,l) - \varepsilon_{tr}(w,l)\big) = 0. \tag{5}$$

For practical purposes this limit is never reached, since only a limited set of training patterns may be available, and even if infinite size training set were available, training on such a set would be limited by time.

Different theoretical approaches provide estimates for the size of the confidence interval on the training error under various settings of the problem of learning from examples. Vapnik–Chervonenkis theory (Vapnik, 1982) is the most comprehensive description of learning from examples. VC-theory provides guaranteed bounds on the difference between the training and generalization error. For regression as well as classification VC theory asserts the results, that with probability larger than $1-\eta$, the inequality

$$\sup_w \big|\varepsilon_{gen}(w,l) - \varepsilon_{tr}(w,l)\big| \leqslant 2\tau \sqrt{\frac{h\big(\ln\frac{2l}{n}+1\big) - \ln\frac{\eta}{9}}{l}} \tag{6}$$

is satisfied. Here $h$ is the VC-dimension of the learning algorithm. The parameter $\tau$ is an upper bound on the error function $\varepsilon$, so $\tau = 1$ for classification task.

From Eq. 6 it follows (Vapnik, 1982) that the asymptotic convergence of the worst-case difference between the generalization and the training error for the realizable task is given by

$$\sup_w \big|\varepsilon_{gen}(w,l) - \varepsilon_{tr}(w,l)\big| \leqslant 2\tau \frac{\ln\frac{2l}{h}}{l/h} \tag{7}$$

for large $l$.

The logarithmic dependency on the ratio $(2l/h)$ present in (7) is removed when the mean case difference between the training and generalization error is investigated instead of the worst-case scenario. For classification tasks, an asymptotic $(h/l)^{\alpha}$ convergence of the difference between the training and generalization error is predicted for selected learning problems, when $l$ is large.

In practice, however, there are several limitations to the mentioned theory. First, the VC-dimension can be analytically described only for a limited number of very simple learning algorithms (i.e., single-layer neural networks). Second, the VC-dimension is in general not constant with respect to $l$ for all learning algorithms. For example, the pruned decision trees have a variable capacity, as opposed to neural networks; thus, the weights $w$ are different for each sample size. Finally, when does $l$ actually become large is unclear and of course not defined. There is no description of behavior of a learning algorithm in so-called "small $l$" regime, neither analytical nor empirical.

## 3. Description of Learning Curve in Small *l* Regime

For this reason, there is a need to formally define the behavior of the learning algorithm in "small $l$" regime. The $(h/l)^{\alpha}$ convergence is predicted for large $l$, but we cannot assume that this function will also best describe the learning curve in the small $l$ regime.

Thus, we need to have a generic description that will conform to the theoretical background on one hand and to the real-life situations on the other. The learning machine is in practice taken as a black box. The way the weights $w$ are calculated and assigned is unknown; it is not necessary that the selection of weights $w$ is optimal. Thus, we can only estimate the theoretical (best) value of $\varepsilon_{gen}(w,l)$. However, when training data are abundant, the difference between generalization and training error diminishes to zero, as in Eq. 5, so the estimate $\varepsilon'_{gen}(l)$ (which is based on training error) is as close to the theoretical $\varepsilon_{gen}(w,l)$ as desired. In the following definitions, we shall outline an estimate for $\varepsilon_{gen(w,l)}$, the $\varepsilon'_{gen}(l)$. First, we will use a generic function $f(l)$, which we define so that it is consistent with the general properties of the theoretical $\varepsilon_{gen}(w,l)$. Next, we add a constant $a$, which enables a better modeling when the $l$ is small. Finally, we add a part that models the noise, which is present in data in the small $l$ regime (e.g., outliers, missing values). Throughout the definitions, we show that such a model is consistent with the theoretical $\varepsilon_{gen}(w,l)$.

DEFINITION 4 (generic generalization error estimation function). Let us define a generic function $f(l)$ that estimates the generalization error based on training size $l$.

$$\varepsilon'_{gen}(l) = f(l), \tag{8}$$

$\varepsilon'_{gen}(l)$ is an estimation of generalization error.

Function $f(l)$ must have the following properties to be consistent (i.e., has the same limit) with the finding that for large $l$ the convergence is described by $(h/l)^{\alpha}$:

$$\lim_{l\to\infty} \varepsilon_{gen}(w,l) = \lim_{l\to\infty} \varepsilon'_{gen}(l) = \lim_{l\to\infty} f(l) = \left(\frac{h}{l}\right)^{\alpha} = 0. \tag{9}$$

Next, the error function should return value 1 (100% error) if no data are available, that is $f(0) = 1$.

Obviously,

$$\sup_{l} f(l) = 1 \tag{10}$$

and

$$\inf_{l} f(l) = 0. \tag{11}$$

**Lemma 1.** *Function $f(l)$ is decreasing.*

*Proof 1.* A function is decreasing if $f(x) \geqslant f(x + \Delta x)$. The $\lim_{l\to\infty} f(l) = 0$ says that for a selected $0 < \delta < 1$ (being as small as desired) there exists an $l$, so that $f(l) < \delta$. If we use $f(0) = 1$ and $f(l) = \delta$, it follows that $\Delta l = l$.

DEFINITION 5 (non-zero generalization error estimation function). Since, in practice, the generalization error rarely reaches zero, we add a constant $a$ to the Eq. 8:

$$\varepsilon'_{gen}(l) = a + f(l). \tag{12}$$

Thus,

$$\lim_{l\to\infty} \varepsilon'_{gen}(l) = a. \tag{13}$$

**Lemma 2.** *We can find such constant $a$ that Function $\varepsilon'_{gen}(l)$ is consistent (has the same limit) with Eq.* 9.

*Proof 2.* From Eq. 9 and Eq. 13 it follows that such a constant exists and that $a = 0$.

The reason why the final error, even in the limit, is not zero may be in the quality of training data. Suppose we have two elements from the set $Z$ (see Definition 3), ($a$, $L_a$) and ($b$, $L_b$) where $a = b$ and $L_a \neq L_b$. It is to note that the database elements are labeled in a wrong way, but this can often happen when the labeling is done by an expert or by a pool of experts. Now, whenever the learning algorithm comes across the first example, the weights are adjusted so that the model will label and $c$, $c = a$, with label $La$. The same will happen with $b$, so $b$ will be misclassified and there will be at least one error. In this case, the error will be at least $1/l$. We may expect that the misclassified data items are evenly distributed throughout the sample, so the function describing the generalization error should have the form as in Eq. 12.

In practice, however, the learning curve is not monotonically decreasing as the sample size increases, especially when $l$ is small. This is caused by the local variance, first observed (but not explained) by Provost *et al.* (1999).

The variation in measurements within the samples (local variance) is resulting from noise variables, which we cannot control directly; we can treat them as a random variable (Cohen, 1995). The noise variables are the consequence of the sampling in small $l$ regime and the structure of the learning algorithms.

The sampling may build the learning set out of the total population so that only some features are included, and others are not, for a given sample size $l_i$. When we increase the sample size to $l_{i+1}$, it can happen that more unseen features are left out as in the sample of size $l_i$, so the performance will decrease. The decrease can be lowered by using statistical approaches such as cross-validation or bootstrapping, but it cannot be totally removed, especially not in the small l regime.

Additionally, some of the learning algorithms may have a fixed capacity, so the whole training set up to the size of $l_i$ is memorized. A relatively low overall generalization error is yielded when using the cross-validation. Then, when the capacity is exceeded, the new features are not included in the model, so the generalization error rises.

These practical observations in a way contradict the requirement that the function describing the generalization error is monotonically decreasing. To model the real life cases and situations more effectively, we need to ease this requirement and at the same conform to the theoretical findings. For this reason we extend the Definition 5 with an addition that will take care of the local variance.

DEFINITION 6 (small $l$ generalization error estimation function). Since we cannot control the noise variables directly, we need to describe their effects. The effects are described by the $\nu(l)$ part in the following equation:

$$\varepsilon'_{gen}(l) = a + f(l) \pm r\cdot \nu(l). \tag{14}$$

The $\nu(l)$ part is actually the deviation of the measurements, defined as

$$\nu(l) = \sqrt{\frac{1}{l-1}\sum_{i=1}^{l}(e_i - \overline{e})^2}, \tag{15}$$

where

$$\overline{e} = \frac{1}{l}\sum_{i=1}^{l} e_i \tag{16}$$

and

$$e_i = f(l_i). \tag{17}$$

The constant $r$ is the range of confidence interval, to be explained later.

**Lemma 3.**

$$\lim_{l\to\infty} a + f(l) \pm \nu(l) = a. \tag{18}$$

*Proof 3.* From Eq. 9 it follows that $\lim_{l\to\infty} f(l) = 0$, and obviously, $\lim_{l\to\infty} a = a$. Thus, we need to prove that $\lim_{l\to\infty} \nu(l) = 0$. We use the Chebychev's Inequality (Bronshtein and Semendjajev, 1987), which implies the weak law of large numbers, i.e.,

$$\forall\varepsilon > 0\colon \lim_{l\to\infty} p\left(\left|\frac{1}{l}\sum_{i=1}^{l} e_i - \overline{e}\right| \geqslant \varepsilon\right) = 0. \tag{19}$$

Thus,

$$\lim_{l\to\infty} \nu(l) = \lim_{l\to\infty}\sqrt{\frac{1}{l-1}\sum_{i=1}^{l}(e_i - \overline{e})^2} = \lim_{l\to\infty}\sqrt{\frac{1}{l}\sum_{i=1}^{l}(e_i - \overline{e})^2}. \tag{20}$$

Since

$$0 \leqslant e_i \leqslant 1 \quad \text{for } \forall i \Rightarrow 0 \leqslant \overline{e} \leqslant 1 \ \text{ for } \forall i \tag{21}$$

it follows:

$$0 \leqslant (e_i - \overline{e})^2 \leqslant |e_i - \overline{e}| \leqslant 1 \quad \text{for } \forall i. \tag{22}$$

Following from Eqs. 19 and 21, we develop Eq. 22 into

$$\lim_{l\to\infty} \nu(l) = \lim_{l\to\infty} \sqrt{\frac{1}{l}\sum_{i=1}^{l}(e_i - \overline{e})^2} \leqslant \lim_{l\to\infty} \sqrt{\left|\frac{1}{l}\sum_{i=1}^{l} e_i - \overline{e}\right|} = 0. \tag{23}$$

By adding the $\nu(l)$ part to the Eq. 14 we have not changed the general description of the learning algorithm, which holds for large $l$. For small $l$, however, we have added the ability to assess the deviation of the error rate. In other words, if we use range $r$ and calculate the deviation, we have a confidence interval in which the error rate can be expected. However, since the number of samples is low, we need to use the confidence intervals for t-tests (which, when the number of samples grows to infinity, converges to confidence intervals for z-test).

The learning curves, if sampled at too high frequency ("too locally"), do not behave well, i.e., the error rate may increase as the sample size $l$ increases (due to irregularities in data). We prove that there exists such a sampling frequency $\Delta l$ so that the graph of the learning curve is always decreasing, i.e., the measured error rates are decreasing and that this is possible if we model the generalization error rate using the Eq. 14.

**Lemma 4.** *It is possible to find such $\Delta l$, so that*

$$\varepsilon'_{gen}(l) \geqslant \varepsilon'_{gen}(l + \Delta l) \quad \textit{for } \forall \Delta l \geqslant l_\delta \geqslant 0. \tag{24}$$

*Proof 4.* We need to prove that (in worst case)

$$\forall \Delta l \geqslant l_\delta\colon\ a + f(l) - r\cdot \nu(l) \geqslant a + f(l + \Delta l) + r\cdot \nu(l + \Delta l) \tag{25}$$

holds. To rewrite,

$$f(l) - r\cdot \nu(l) \geqslant f(l + \Delta l) + r\cdot \nu(l + \Delta l). \tag{26}$$

Since, by the definition of a limit,

$$\lim_{x\to\infty} f(x) = L \iff \forall \varepsilon > 0\colon \exists \delta > 0\colon x > \delta \Rightarrow \big|f(x) - L\big| < \varepsilon \tag{27}$$

and since, by Proof 3,

$$\lim_{x\to\infty} \nu(x) = 0 \tag{28}$$

we can select

$$\varepsilon_r > 0\colon\ \exists\delta_{l_r} > 0 \Rightarrow \left|\nu(x)\right| < \varepsilon_r \tag{29}$$

or analogously,

$$\begin{aligned}&\varepsilon_\nu > 0\colon\ \exists\delta_{l_\nu} > 0 \Rightarrow \left|\nu(x)\right| < \frac{\varepsilon_r}{r},\\ &\left|\nu(x)\right| < \varepsilon_\nu\Big|\frac{\varepsilon_r}{r} = \varepsilon_r.\end{aligned} \tag{30}$$

Additionally, since – from Eq. 9

$$\lim_{x\to\infty} f(x) = 0, \tag{31}$$

we can select

$$\varepsilon_f > 0\colon\ \exists\delta_{l_f} > 0 \Rightarrow \left|f(x)\right| < \varepsilon_f. \tag{32}$$

If we select $\delta_l = \mathrm{MAX}(\delta_{l_f}, \delta_{l_\nu})$, we can rewrite Eq. 25 as

$$f(l) - r\cdot\nu(l) > \varepsilon_f + \varepsilon_\nu. \tag{33}$$

Since the expression $f(l) - r\cdot\nu(l)$ describes the error rate, it has to be positive, i.e.,

$$\forall l\colon\ f(l) - r\cdot\nu(l) > 0. \tag{34}$$

We can always calculate the values of functions $f(l)$ and $\nu(l)$, i.e.:

$$f(l) = \alpha, \quad r\cdot\nu(l) = \beta. \tag{35}$$

Now, we can always select such small $\varepsilon_f$ and $\varepsilon_\nu$ so that

$$\forall\delta_l\colon\ \alpha - \beta > \varepsilon_f + \varepsilon_\nu. \tag{36}$$

From $\delta_l$ we can calculate

$$\Delta l = \delta_l - l. \tag{37}$$

From Lemma 4 it follows that we can develop such a sampling schedule so that eventually the error rate drops for every sample size, from a given sample size on. This means that once we have the monotonically decreasing error rate, the graph – learning curve – is decreasing. Thus, the learning curve starts to "behave well" and the principal component in Eq. 14, the $f(l)$, describes the general behavior of the learning algorithm, without the impact of the variance in local measurements.

## 4. Foundations for Practical Observation of a Model's Performance

The formal model for description of learning in the small $l$ regime was described in the previous section. Here, we develop a foundation for the tool with which we can do practical observations of the algorithm's performance based on small samples, i.e., in small $l$ regime (Brumen *et al.*, 2004). First, we need to observe the algorithm's performance by measuring the error rate $(e)$ versus the sample size $(l)$. To observe the performance, we need to sample the data from the database. The sampled data need to be manually prepared by an expert or a group of experts. Once the data are properly prepared, the algorithm is used and the model based on the prepared data is built. The model's performance is measured, and the error rate for the given sample size is obtained. The procedure needs to be repeated so that several points (error rate, sample size) are available and the graph of the learning curve can be plotted. When the graph has the proper shape (i.e., the learning curve is well-behaved), the estimate for the future performance can be calculated. Finally, when the estimate is close enough to the real values (or the resources are exhausted), the process can terminate.

Simple $k$-fold cross validation procedure tests the performance of a classifier after a given amount of training. To chart a learning curve, however, it is desirable to measure the performance repeatedly as the amount of training is increased. For this purpose, Lehnert and McCarthy (Lehnert *et al.*, 1993) introduced a variant of a cross validation, called *incremental $k$-fold cross-validation*, which is a widely adopted method. Their method, however, has a fixed, known number of iterations (because the size of the data set is known). We modify the McCarthy's and Lehnert's procedure (Brumen *et al.*, 2001b; Brumen *et al.*, 2002b) to be able to cope with the fact that in general, the database size is not known in advance, and that only the current sample size is known. Additionally, we want to use as many items as possible in the learning phase, and especially the testing phase. When we have a large data set, we need to analyze the performance of the algorithm on-line, so that we can stop the process if the costs are exceeded, or the performance is on the appropriate level.

For this reason we developed an adaptive incremental $k$-fold cross-validation method (Procedure 1), which takes into the account the requirements set in the framework.

The main modification we made to the incremental $k$-fold cross-validation is the adaptive increment and the stopping criteria. We still get to see the cumulative effects of training. The procedure repeatedly ($k$ times) trains a classifier on $n - n/k$ items, then tests it on $n/k$ from the test set (and records the $i$th performance). The overall performance on $n$ items is calculated by averaging the $k$ intermediate results.

Afterwards, the size of the *original* sample is incremented by a factor, which is calculated *adaptively*, based on the properties of the performance curve built in run-time. This new sample is composed of old items (used in the previous iteration) plus some new ones, picked randomly from the samples. We added some "new knowledge" to the knowledge base. When the classifier is run on the new sample, we get to see a cumulative effect of the new knowledge on the performance. The effect can be either positive (e.g., the error rate decreases) or negative (e.g., the error rate increases).

Procedure 1. Adaptive incremental $k$-fold cross-validation.

```
Repeat
  1. Shuffle the items in the training set.
  2. Divide the training set into k equal parts of size n
  3. Do i = 1 to k times:
      a. Call the ith set of n samples the test set and put it
         aside.
      b. Train the system on the remaining k - 1 sets; test the
         system on the test set, record the performance.
      c. Clear memory, that is, forget everything learned during
         training.
  4. Calculate the performance of training averaged over the
     k test sets.
  5. Increment the size of the training set (i.e., add additional
     data to the existing training set) based on the properties
     of the performance curve
     Until (performance is satisfactory) or (costs are exceeded)
```

In the adaptive incremental $k$-fold cross validation procedure the performance is calculated each time on different training *and* test set, since both are increased in size from the previous step. Thus, the error rate is more accurate from step to step because more items are available to check the performance of the learning algorithm.

The adaptive incremental cross-validation is an improved version of a standard incremental cross validation because we do not need to know in advance the exact number of items in a sample. This is especially advantageous when the items for the sample need to be prepared or pre-processed. Additionally, we can observe the cumulative effect of training on-line (after each set of $k$-fold cross-validations), instead of at the end of the whole incremental cross-validation procedure. Our approach increases the test (and training) sample size in every step, while the original one keeps the test size constant, and only increases the training sample size.

The learning curve depicts the performance of the learning algorithm. Observation of the learning curve is a very important task, because the learning curve explains the behavior of the learning algorithm. Based on the properties of the learning curve, the adaptive incremental $k$-fold cross validation loops. When the loop conditions are met, the measurement of the points of the learning curve stops and the points can be used to build the estimation model for the curve. In this sub-section, we develop and devise the loop conditions.

In the step of observing the learning curve, the shape and the quality of the learning curve is monitored. The most important task is to detect the point where the learning curve starts to behave well. Since we do not have any analytical function to observe (the description of which, by the way, we want to obtain as soon as possible), we need to observe the graph of the points that are (later) used to build the analytical model of the learning curve.

We found out empirically that in many cases the error-rate learning curve starts behaving well when its graph becomes monotonically decreasing – see Eq. 38 – and concave up – see Eq. 39 – for a given number of points. In case the learning curve depicts the accuracy, it is monotonically increasing and concave down.

$$y_{i-2} > y_{i-1} > y_i, \tag{38}$$

$$\frac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}} < \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{39}$$

for $2 \leqslant i \leqslant m.$

The task in the step of observing the learning curve is to observe whether the conditions of discrete concavity are met. If they are, the estimation step is performed. In the estimation step, we estimate the error rate for the next step and the "final step". After the estimation is made, the actual error rate, $e_i$, the estimated error rate for the next step $e_{next}$ and the estimated error rate for a large data size, $e_{large}$ are compared. If they are all within a limit (provided by the operator), we can conclude that the performance of the algorithm will not improve drastically. To formalize, we check that the Eq. 40 holds:

$$\text{MAX}\big(|e_i - e_{next}|, |e_{next} - e_{large}|, |e_i - e_{large}|\big) < \varepsilon. \tag{40}$$

If the criteria are not met, the amount of data is adjusted accordingly and the model building and the performance measurement step loops once again – if the resources are not exhausted.

The above equations are very important because they constitute the decision part of the adaptive incremental $k$-fold cross-validation, making it distinct from the other cross-validations. These equations enable the proactive development of the learning curve model: hereby we answer the question "When to stop measuring the learning curve points?" When the measurements stop (based on the properties of the learning curve, of course), the model can be calculated (fitted) from the existing points and the future performance can be assessed for an arbitrary sample size.

## 5. Experiment and Results

### 5.1. *Data Source and Equipment Used*

The approach was used and tested on five data sets – one from UCI Knowledge Discovery in Databases Repository, and four from UCI Machine Learning Repository (Blake and Merz, 1998). The datasets were chosen following these criteria that the datasets must be public, must be of various sizes and from different domains. For the clarity of presentation, the results are shown only for one data set ("Adult"); the results for others are available in (Brumen, 2004).

We have used the See5 classification algorithm by Quinlan (Quinlan, 2000) on a Windows-based PC. The reason we chose this algorithm is that it is freely available for time limited testing and because it is an improved version of the most popular classification algorithm C4.5. Additionally, it has cross-validation built-in.

We have built a prototype and implemented the method of the adaptive incremental approach. For calculation of the models' parameters, we used Matlab's `LSQNONLIN` function from the Optimization Toolbox (Coleman *et al.*, 1999).

## 5.2. *Validity of Results*

The results are statistically validated using built-in $k$-fold cross validation method and the $r\sigma$ confidence intervals.

The $k$-fold cross-validation method assures that each point $(l_i, e_i)$ in the learning curve is statistically valid and that it truly represents the error rate for *any* sample of size $l_i$.

The $r\sigma$ confidence intervals assure that the estimated error rate is statistically valid, meaning that the difference between the calculated (estimated) value and the actual value is insignificant.

The probability threshold for rejecting the null hypothesis (i.e., the measured error rate lies within the confidence interval) was chosen to be 0.001. Namely, the Bonferroni adjustment requires that the probability of falsely rejecting the null hypothesis need to be decreased by the number of measurements (Cohen, 1995), which in our case was 25. The standard probability for a single test is $p = 0.05$ (divided by 25 yields approximately 0.002; due to unavailability of the t-tables we selected a more restrictive threshold of 0.001).

Since no approach that is like ours or even distantly similar was ever developed, we do not include any comparisons of the results to other results.

## 5.3. *Application of the Approach to the Dataset*

The size of the data set was $L = 48842$ instances. We decided for the following amounts of data to be used by the learning algorithm {80, 90, 100, 200, ..., 1000, 2000, ..., 10000, 20000, ..., 48842} and the $\varepsilon$ in Eq. 40 was set to 0.02 (2%). The final error rate was estimated for $l = 500000$, approximately 10-times the number of the available instances.

For each sample size we measured the error rate. We continued to do so until the conditions for Eq. 38 and Eq. 39 were fulfilled. This first happened at $l_9 = 700$. At this point, we checked also the conditions of Eq. 40, which were not met. The second time the conditions of the above mentioned equations were met at $l_{14} = 3000$. Again, the Eq. 40 was not met (MAX(...) = 3.43%). The first time all three conditions were met was at $l_{19} = 8000$ (shaded area of the Table 1). At this point, the error rate $e_{19}$ was 15.29%; at the two previous points, the error rates were $e_{17} = 15.96\%$ and $e_{18} = 15.34\%$, respectively. At this point, the calculated parameters for the full learning curve were $a = 14.1066$, $b = 110.7611$ and $c = -0.5000$. The model of the full learning curve built at $l_{19} = 8000$ was thus:

$$e = 14.1066 + 110.7611 \cdot l^{-0.5} \pm 1.464. \tag{41}$$

In general, the process would stop at $l_{19} = 8000$. Of course, this way we would not know how good our model is, how well it describes the behavior of the model if built by

Table 1

Measurements and estimations of error rate

| **Step** ($i$) | **Data size** ($l_i$) | **Error rate** ($e_i$) | **Next error rate** ($e_{next}$) | **Final error rate** ($e_{large}$) | **Conf. interval** | $\pm r\sigma$ |
|---|---|---|---|---|---|---|
| 1 | 80 | 26.6400 | 25.9408 | 17.6602 | (NA,NA) | NA |
| 2 | 90 | 26.8500 | 26.0597 | 19.8996 | (NA,NA) | NA |
| 3 | 100 | 27.6000 | 25.4492 | 22.5316 | (NA,NA) | NA |
| 4 | 200 | 24.0000 | 23.6058 | 18.8386 | (9.50,41.40) | 24.579 |
| 5 | 300 | 17.4000 | 19.3185 | 11.4405 | (-0.97,48.19) | 44.540 |
| 6 | 400 | 20.4800 | 19.4606 | 13.0881 | (-25.2,63.86) | 22.651 |
| 7 | 500 | 19.3400 | 18.9213 | 12.9560 | (-3.19,42.11) | 9.279 |
| 8 | 600 | 17.5300 | 18.0219 | 12.0255 | (9.64,28.20) | 8.041 |
| 9 | 700 | 16.7800 | 17.3713 | 11.4906 | (9.98,26.06) | 6.634 |
| 10 | 800 | 15.6600 | 16.7229 | 10.9075 | (10.74,24.01) | 4.499 |
| 11 | 900 | 17.5700 | 16.7923 | 11.4885 | (12.22,21.22) | 4.402 |
| 12 | 1000 | 18.0500 | 15.4604 | 11.5095 | (12.39,21.19) | 5.610 |
| 13 | 2000 | 17.2500 | 15.1318 | 11.9376 | (9.85,21.07) | 1.739 |
| 14 | 3000 | 17.0300 | 15.4753 | 13.6003 | (13.39,16.87) | 2.266 |
| 15 | 4000 | 17.0300 | 15.6261 | 14.2090 | (13.21,17.74) | 0.526 |
| 16 | 5000 | 15.7300 | 15.4869 | 14.2053 | (15.10,16.15) | 3.057 |
| 17 | 6000 | 15.9600 | 15.4482 | 14.2832 | (12.43,18.54) | 2.785 |
| 18 | 7000 | 15.3400 | 15.3412 | 14.2580 | (12.66,18.23) | 1.243 |
| 19 | 8000 | 15.2900 | 15.2741 | 14.2633 | (14.10,16.58) | 1.464 |
| 20 | 9000 | 15.4300 | 15.2267 | 14.2768 | (13.81,16.74) | 0.275 |
| 21 | 10000 | 15.0400 | 14.8859 | 14.2589 | (14.95,15.50) | 0.761 |
| 22 | 20000 | 14.5000 | 14.7130 | 14.2285 | (14.13,15.65) | 1.784 |
| 23 | 30000 | 13.9000 | 14.5471 | 14.1390 | (12.93,16.50) | 2.162 |
| 24 | 40000 | 13.8500 | 14.3771 | 13.9926 | (12.38,16.71) | 1.363 |
| 25 | 48842 | 13.8700 | 13.9310 | 13.8935 | (13.01,15.74) | 0.094 |

using all the data available. For this reason we continued with the process. The results are shown in Table 1.

The full learning curve that was measured and modeled (at $l_{19} = 8000$) is depicted in Fig. 2.

It is interesting to notice that at $l_6 = 400$ the estimated final error rate of $13.0881\%$ was already within $1\%$ of the true final error rate of $13.87\%$ and also within $1\%$ with the final estimate of 13.931 ($\pm r\sigma = 0.064824$) at $l_{25} = 48842$. The estimation at $l_6 = 400$ was calculated with $0.81\%$ of the total amount of the available data. At $l_{19} = 8000$, or $16.4\%$ of the total amount of available data, we were able to build a model that estimated the final error that differed for $0.3933\%$ from the actual final error rate at $l_{25} = 48842$, and the difference to the final estimate of 13.931 was $0.3323\%$.

Throughout the building of the model, we were calculating the estimated error rate for the next step. For example, the error rate at $l_{15} = 4000$ was $17.03\%$. The next sample

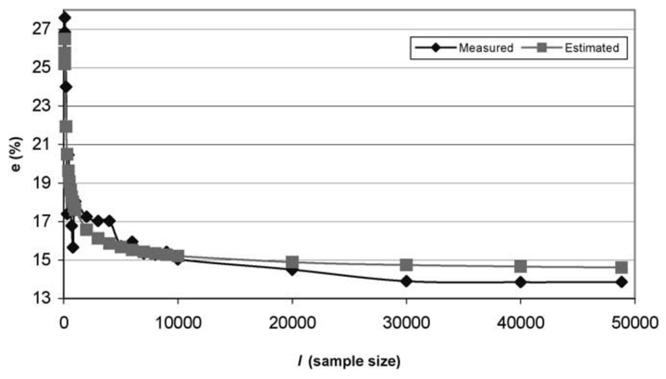

Fig. 2. Full learning curve for adult dataset (measured and estimated).

size in the schedule was $l_{16} = 5000$. At $l_{15} = 4000$ the estimated error rate for the next step was 15.6261% ($\pm 0.526$) and the actual error rate was 15.73%.

In fact, the average difference between the estimations for error rate at the next step and the later measured actual error rate is 1.1067%, with standard deviation of 1.2436%. The maximum difference was 6.2% at $l_{14} = 3000$, the second largest difference was 1.8982% at $l_{13} = 2000$, and the minimum was 0.0512% at $l_{18} = 7000$. At all points except one (at $l_{21} = 10000$), the estimations were accepted as significant because the values were within the $r\sigma$ interval.

## 6. Conclusion

The problem we have solved in the paper was to assess and estimate the future behavior (i.e., performance in terms of error rate) of a learning algorithm based on observation of its performance on smaller sample sizes. The solution of the problem was formally developed; we presented the design (method) of the solution. The developed adaptive incremental cross-validation is an improved version of a standard incremental cross validation because we do not need to know in advance the exact number of items in a sample. Additionally, we can observe the cumulative effect of training on-line, instead of at the end of the whole incremental cross-validation procedure. Our approach increases the test (and training) sample size in every step, while the original one keeps the test size constant, and only increases the training sample size.

The contribution of the paper is a new understanding of learning in small $l$ regime and a new method for assessment of classification algorithms already in the early stages of the learning. In Section 5, we reported the results on one of the publicly available data sets and the results confirm the validity of the approach. Namely, the model of a full learning curve was built on smaller sample size and used to estimate the performance on a larger sample size. When the actual (larger) sample size was used, we compared the measured and the calculated (estimated) results. The differences are statistically insignificant. Thus, the model and the approach are valid and correct.

The method is general and useful in sense that it can be applied to any classification project. A typical scenario, where the results of the paper can be used, is a data mining problem, where the task is to build a model based on data. The data miner can use our approach to assess the performance of the algorithm used to build the model, both in the early stage when the data are scarce (and the future performance is sought), and at the end, when no more data are available, and the question is, whether the performance could improve any further. Thus, by using the developed method, the results in any stage of data mining can be extended so the insight into algorithm's future performance is gained.

## References

Anderson, J.R., and L.J. Schooler (1991). Reflections of the environment in memory. *Psychological Science*, **2**(6), 396–408.

Berry, M.A.J., and G. Linoff (1997). *Data Mining Techniques for Marketing, Sales, and Customer Support*. John Wiley & Sons, Inc., New York.

Blake, C.L., and C.J. Merz (1998). *UCI Repository of Machine Learning Databases*. Department of Information and Comp. Sci., University of California, Irvine, CA, USA. `http://www.ics.uci.edu/~mlearn/LRepository.html`.

Bronshtein, I.N., and K.A. Semendjajev (1997). *Handbook of Mathematics*, 3rd edition. Springer Verlag.

Brumen, B., T. Welzer, I. Golob and H. Jaakkola (2001a). Convergence detection in classification task of knowledge discovery process. In *PICMET '01, Portland International Conference on Management of Engineering and Technology. Proceedings*, vol. 1. *Book of Summaries at PICMET'01* (IEEE Cat. No.01CH37199). pp. 66.

Brumen, B., H. Jaakkola and T. Welzer (2001b). Prediction sample size in data mining tasks using additive incremental approach. *Frontiers in Artificial Intelligence and Applications*, **67**, 246–270.

Brumen, B., I. Golob, T. Welzer, I. Rozman, M. Družovec and H. Jaakkola (2002a). Data protection for outsourced data mining. *Informatica* (Ljublj.), **26**(2).

Brumen, B., T. Welzer, H. Jaakkola, I. Golob and I. Rozman (2002b). Towards cost-effective construction of classification methods. *Frontiers in Artificial Intelligence and Applications*, **73**, 301–306.

Brumen, B., I. Golob, T. Welzer, I. Rozman, M. Družovec, H. Jaakkola (2003). An algorithm for protecting knowledge discovery data. *Informatica* (Lithuania), **14**(3).

Brumen, B. (2004). Empirical procedure for assessment of classification algorithms. *Doctoral Dissertation*, University of Maribor, Slovenia.

Brumen, B., I. Golob, H. Jaakkola, T. Welzer and I. Rozman (2004). Early assessment of classification performance. *Australasian CS Week Frontiers*, **32**, 91–96.

Brumen, B., T. Welzer, M. Družovec, I. Golob, H. Jaakkola, I. Rozman and J. Kubalík (2005a). Protecting medical data for decision-making analyses. *Journal of Medical Systems*, **29**(1), 65–80.

Brumen, B., T. Welzer, I. Rozman, M. Hölbl and H. Jaakkola (2005b). Convergence detection criteria for classification based on final error rate. In: I.J. Rudas (Ed.), *Proceedings of the IEEE 3rd International Conference on Computational Cybernetics*. pp. 41–46.

Cabena, P., P. Hadjinijan, R. Stadler, J. Verhees and A. Zanasi (1997). *Discovering Data Mining*: *From Concept to Implementation*. Prentice Hall Ptr, New Jersey, USA.

Cohen, P.R. (1995). *Empirical Methods for Artificial Intelligence*. Mit Press, Cambridge, MA, USA.

Coleman, T., M.A. Branch and A. Grace (1999). *Optimization Toolbox for Use with Matlab®*. User's Guide, Version 2. Mathworks Inc.

Frey, L.J., and D.H. Fisher, Jr. (1999). Modeling decision tree performance with the power law. In D. Heckerman and J. Whittaker (Eds.), *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. San Francisco, Morgan Kaufmann, Inc., USA.

Harris-Jones, C., and T.L. Haines (1997). Sample size and misclassification: Is more always better? *Working Paper*, AMSCAT-WP-97-118, AMS Center for Advanced Technologies.

Lehnert, W.G., J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F.F. Feng, C. Dolan and S. Goldman (1993). Umass/Hughes: description of the CIRCUS system as used for MUC-5. In *Proceedings of 5th Message Understanding Conference*. MKP Inc., San Mateo, USA. pp. 277–291.

Provost, F., D. Jensen and T. Oates (1999). Efficient progressive sampling. In S. Chaudhuri and D. Madigan (Eds.), *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, August 15–18, San Diego. ACM Press. pp. 23–32.

Quinlan, R.J. (2004). *Data Mining Tools See5 & C5.0.* `http://www.rulequest.com/` (last visited 8/2004).

Vapnik, V.N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, USA.

Witten, I.H., and E. Frank (2000). *Data Mining*: *Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, San Francisco, USA.

**B. Brumen** is an assistant professor (part time) at Faculty of Electrical Engineering and Computer Science, University of Maribor and secretary general of the same university (full time). He received a doctor's degree (computer science) in 2004. His research interests include data mining, data security, data analyses, and data reusability. As a member of Database Technologies Laboratory he actively participates in several international and national projects, related to data issues. He cooperates with researchers at Tampere University of Technology since 1999. The results were published at several international conferences and in journals.

**M.B. Jurič** is an associate professor at Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests are in the area of application integration – Enterprise Application Integration (EAI), Business to business and business to consumer integration, Web services, Component models, Patterns for integration, Performance analysis and optimization. He received his doctor's degree in 1999 at University of Maribor.

**T. Welzer** is a professor at Faculty of Electrical Engineering and Computer Science, University of Maribor. She graduated in 1984, received master's degree (computer science) in 1989 and doctor's degree (computer science) in 1995. Her research interests include data modelling, data technologies, data reuse and medical informatics.

**I. Rozman** is a professor at Faculty of Electrical Engineering and Computer Science, University of Maribor. He graduated in 1977 (electrical engineering), received master's degree (computer science) in 1980 and doctor's degree (computer science) in 1983. His research interests include integration of information systems, software metrics and systems quality. Currently, he serves the second term as a rector of University of Maribor.

**H. Jaakkola** is a professor of software engineering at Tampere University of Technology, director of Center of Software Expertise (CoSE) and head of the Regional Institute of Tampere University of Technology in Pori. His research interests cover software engineering (software process improvement and object technologies) and technology management (technology diffusion and transfer). Professor Hannu Jaakkola has received doctor's degree (engineering) at Tampere University of Technology in 1991, BSc (business economics) at University of Tampere in 1979 and master's degree (computer science) at University of Tampere in 1974.

**A. Papadopoulos** received his 5-year diploma degree in computer engineering and informatics from the University of Patras and his PhD degree from Aristotle University of Thessaloniki in 1994 and 2000 respectively. He has published several research papers in journals and proceedings of international conferences. From March 1998 to August 1998 he was a visitor researcher at INRIA Research Center in Rocquencourt, France, to perform research in spatial databases. His research interests include databases, data stream processing, data mining and information retrieval. His research work has over 250 citations in scientific journals and conference proceedings. He has served as a track co-chair of ACM SAC DTTA (Database Technologies Techniques and Applications) Track 2005, 2006 and 2007. He is a member of the Technical Chamber of Greece. Currently, he is a lecturer in the Department of Informatics of Aristotle University of Thessaloniki.

## Klasifikavimo modelių įvertinimas naudojant mažą duomenų kiekį

Boštjan BRUMEN, Matjaž B. JURIČ, Tatjana WELZER, Ivan ROZMAN,
Hannu JAAKKOLA, Apostolos PAPADOPOULOS

Vienas iš duomenų gavybos uždavinių yra klasifikavimas, atliekantis požymių (stebėjimų) atvaizdavimą į iš anksto apibrėžtas klases. Klasifikavimo modeliai konstruojami naudojant esamus duomenis. Iš principo modeliai, sukonstruoti naudojant daugiau duomenų, duos geresnius rezultatus. Tačiau ryšys tarp duomenų ir modelio veikimo dar nėra gerai suprastas, išskyrus tai, kad klasifikavimo metodų tikslumas mažėja mažėjant duomenų kiekiui. Straipsnyje pateikiama ankstyvojo gautų žinių įvertinimo metodika. Ši metodika remiasi informacija apie metodų veikimą esant mažai duomenų apimčiai. Metodas formaliai apibrėžtas, eksperimentai parodė jo naudingumą.