

# Early Assessment of Classification Performance<sup>1</sup>

Boštjan Brumen<sup>°</sup>, Izidor Golob<sup>°</sup>, Hannu Jaakkola<sup>°</sup>, Tatjana Welzer<sup>°</sup>, Ivan Rozman<sup>°</sup>

<sup>°</sup>University of Maribor, Faculty of Electrical Engineering, Computer Science and Informatics  
Smetanova 17, SI-2000 Maribor, Slovenia

<sup>2</sup>Tampere University of Technology, Pori School of Technology and Economics  
Pohjoisranta 11, PO BOX 300, FIN-28101 Pori, Finland

<sup>°</sup>{bostjan.brumen|izidor.golob|aida.kamisalic|welzer|i.rozman}@uni-mb.si

<sup>°</sup>hannu.jaakkola@pori.tut.fi

## Abstract

The ability to distinguish between objects is the fundamental to learning and intelligent behavior in general. The difference between two things is the information we seek; the processed information is actually the base for the knowledge. Automatic extraction of knowledge has been in interest ever since the advent of computing, and has received a wide attention with the successes of data mining. One of the tasks of data mining is also classification, which provides a mapping from attributes (observations) to pre-specified classes. Based on the distinction between the objects they are mapped into different classes.

In the paper, we present an approach for early assessment of the extracted knowledge (classification models) in the terms of performance (accuracy). The assessment is based on the observation of the performance on smaller sample sizes. The solution is formally defined and used in an experiment. The results confirm the correctness of the approach.

*Keywords:* assessment, classification, accuracy, learning curve.

## 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, 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, 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 (Groth, 1998), (Cabena, 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 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 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.

*Paper organization.* In the following sub-section give a brief review of the related work and outline the open research problem. Next, we emphasize the contribution of the paper. In Section 2, we describe a formal setting for the solution of the problem and in Section 3 we develop a method based on the formal description. In Section 4 we describe our experiment and the results. We conclude the paper with final remarks in Section 5.

---

<sup>1</sup> Copyright © 2004, Australian Computer Society, Inc. This paper appeared at *ACSW Frontiers 2004, Dunedin, New Zealand*. Conferences in Research and Practice in Information Technology, Vol. 32. James Hogan, Paul Montague, Martin Purvis and Chris Steketee, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

## 1.1 Related work

When measuring the accuracy, we build a so-called learning curve. The learning curve depicts the relationship between sample size and model performance (see Figure 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 of the model produced by an algorithm when given a training set of size  $n$ .

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, 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.

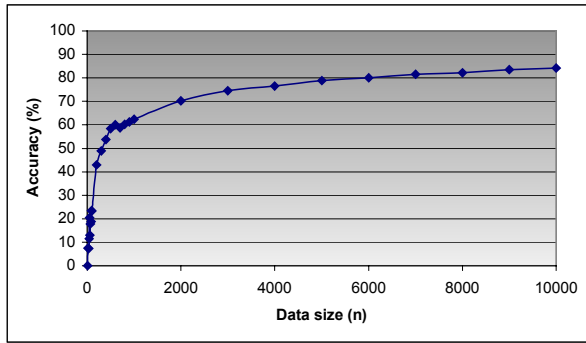


Figure 1: A typical learning curve

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 (VC-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

$$\sup_w |\mathcal{E}_{gen}(w, l) - \mathcal{E}_{tr}(w, l)| \leq 2\tau \sqrt{\frac{h \cdot (\ln \frac{2l}{h} + 1) - \ln \frac{\eta}{9}}{l}} \quad (1)$$

is satisfied. Here  $h$  is the VC-dimension of the learning algorithm. The parameter  $\tau$  is an upper bound on the error function  $\mathcal{E}$ , so  $\tau=1$  for classification task;  $w \in \mathcal{R}^m$  and  $l$  is the sample size.

From (1) 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:

$$\sup_w |\mathcal{E}_{gen}(w, l) - \mathcal{E}_{tr}(w, l)| \leq 2\tau \frac{\ln \frac{2l}{h}}{l/h} \quad (2)$$

for large  $l$ . The logarithmic dependency on the ratio  $(2l/h)$  present in (2) 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 (Vapnik, 1982).

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. Additionally, worst-case emphasis in the approach makes it unusable in practice. 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.

## 1.2 Paper contribution

In the paper, we give a method for 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.

## 2 Formal Setting for Early Assessment of Classification Performance

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, 1997), (Frey, 1999)),

However, 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 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  $\mathcal{E}_{gen}(w, l)$ . However, when training data are abundant, the difference between generalization and training error diminishes to zero (Vapnik, 1982), so the estimate of the performance (which is based on the training error) is as close to the theoretical  $\mathcal{E}_{gen}(w, l)$  as desired. In the continuation, we shall outline an estimate for  $\mathcal{E}_{gen}(w, l)$ ,

the  $\mathcal{E}'_{gen}(l)$ , which is to be obtained by measurements.

First, we use a generic function  $f(l)$ , which we define so that it is consistent with the general properties of the theoretical description of the generalization error:

$$\mathcal{E}'_{gen}(l) = f(l) \quad (3)$$

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  $(h/l)^\alpha$ :

$$\lim_{l \rightarrow \infty} \varepsilon_{gen}(w, l) = \lim_{l \rightarrow \infty} \varepsilon'_{gen}(l) = \lim_{l \rightarrow \infty} f(l) = \left(\frac{h}{l}\right)^\alpha = 0 \quad (4)$$

Since, in practice, the generalization error rarely reaches zero, we add a constant  $a$ :

$$\varepsilon'_{gen}(l) = a + f(l) \quad (5)$$

The reason why the final error, even in the limit, is not zero may be in the quality of training data. The database elements can be labeled in a wrong way. This situation can often happen when the labeling is done by a human expert or by a pool of experts. Suppose we have two data items with value  $a$ ; the first is labelled with label  $La$ , the second (wrongly) with  $Lb$ . Whenever the learning algorithm comes across the first example, the weights are adjusted so that the model will label any data item having value  $a$  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.

Finally, we add a part that models the noise, which is present in the small  $l$  regime.

$$\varepsilon'_{gen}(l) = a + f(l) \pm r \cdot v(l) \quad (6)$$

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. For example, 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.

The  $v(l)$  part is actually the deviation of the measurements, defined as:

$$v(l) = \sqrt{\frac{1}{l-1} \sum_{i=1}^l (e_i - \bar{e})^2} \quad (7)$$

where

$$\bar{e} = \frac{1}{l} \sum_{i=1}^l e_i \quad (8)$$

and

$$e_i = f(l_i) \quad (9)$$

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

By adding the  $v(l)$  part to the (6) 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 expected error rate will be.

### 3 Observation of 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 method for obtaining the insight into the algorithm's performance based on small samples (in small  $l$  regime).

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 observed. 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 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, 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 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.

**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)
```

Unlike the *incremental k-fold cross-validation* procedure, this one tests an algorithm for an unknown number of iterations. The number of iterations is defined *adaptively*, based on the properties of the performance curve built in run-time (as described in step 5). 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  $i^{\text{th}}$  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).

In the adaptive incremental k-fold cross validation 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 every time, while the original ones keeps the test size constant, and only increases the training sample size.

**3.1 Observation of the learning curve**

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 model. 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. 10) and concave up (see Eq. 11) for a given number of points. In case the learning curve depicts the accuracy, it is monotonically increasing and concave down.

$$y_{i-2} > y_{i-1} > y_i \tag{10}$$

$$\frac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}} < \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{11}$$

for  $2 \leq i \leq m$ .

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. (12) holds:

$$\text{MAX} (|e_i - e_{next}|, |e_{next} - e_{large}|, |e_i - e_{large}|) < \varepsilon \tag{12}$$

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.

## 4 Experiment and results

### 4.1 Data Source and Equipment Used

The approach was used and tested on three data sets – one from UCI Knowledge Discovery in Databases Repository, and two from UCI Machine Learning Repository (

MLArchive, 1998). The datasets were chosen following these criteria that the datasets must be public, must be of various sizes and from different domains. Due to the lack of space, the results are shown only for one data set (“Adult”); the results for others are available in (Brumen, 2001).

We have used the See5 classification algorithm by Quinlan (Quinlan, 2000) on a Windows98 - based PC (128 Mb RAM, Intel Celeron 333 Mhz). 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, 1999).

### 4.2 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 (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 close to 30. The standard probability for a single test is  $p=0.05$  (divided by 30 yields approximately 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.

### 4.3 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  $\varepsilon$  (Eq. 12) 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. 10 and Eq. 11 were fulfilled. This first happened at  $l_9=700$ . At this point, we checked also the conditions of Eq. 12, which were not met. The second time the conditions of the above mentioned equations were met at  $l_{14}=3000$ . Again, the Eq. 12 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:

$$e = 14.1066 + 110.7611 \cdot l^{-0.5} \pm 1.464 \quad (13)$$

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

Table 1: Measurements and estimations of error rate

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

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

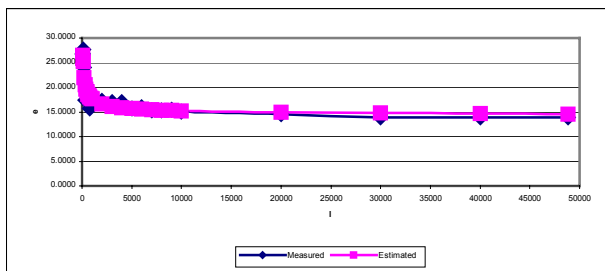


Figure 2: Full Learning Curve for Adult Dataset (Measured and Estimated)

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.

## 5 Conclusion

The problem we have solved in the paper was to assess the future behavior (performance) of a learning algorithm based on observation of its performance on smaller sample sizes. The solutions are novel since no similar solutions or approaches exist. The solution of the problem was formally developed; we presented the design (method) of the solution. In Section 4, we report the results on one of the publicly available data sets.

We have proven that our approach is valid, accurate and correct: the model of full learning curve was built on smaller sample size and used to estimate the performance on 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.

## 6 References

Anderson, 1991: Anderson, John R.; Schooler, Lael J.: Reflections of the environment in memory, *Psychological Science*, Vol. 2, No. 6, pp. 396-408, 1991

Berry, 1997: Berry, Michael A. J.; Linoff, Gordon: *Data Mining Techniques for Marketing, Sales, and Customer Support*, John Wiley & Sons, Inc., New York, 1997

Brumen, 2001: Brumen, Boštjan; Welzer, Tatjana, Golob, Izidor; Jaakkola, Hannu: Convergence detection in classification task of knowledge discovery process. In: Kocaoglu, Dundar F.; Anderson, Timothy R. Eds.: *Portland international conference on management of engineering and technology, Portland, Oregon - USA, July 29 - August 2, 2001 : proceedings. Vol. 2, Papers presented at PICMET'01*. Portland: Portland State University, 2001

Cabena, 1997: Cabena, Peter; Hadjinijan, Pablo; Stadler, Rolf; Verhees, Jaap; Zanasi, Alessandro: *Discovering Data Mining: From Concept to Implementation*, Prentice Hall Ptr, New Jersey, USA, 1997

Cohen, 1995: Cohen, Paul R.: *Empirical Methods for Artificial Intelligence*, Mit Press, Cambridge, MA, USA, 1995

Coleman, 1999: Coleman, Thomas; Branch, Mary Ann; Grace, Andrew: *Optimization Toolbox for use with Matlab®*, User's Guide Version 2, Mathworks Inc, 1999.

Frey, 1999: Frey, Lewis J.; Fisher, Douglas H. Jr.: Modeling Decision Tree Performance with the Power Law, In Heckerman, David; Whittaker, Joe; Eds.: *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, San Francisco, Morgan Kaufmann, Inc., USA, 1999

Groth, 1998: Groth, Robert: *Data Mining: A Hands-On Approach for Business Professionals*, Prentice Hall PTR, Upper Saddle River, New Jersey, USA, 1998

Harris-Jones, 1997: Harris-Jones, Chirs; Haines, Troy L.: Sample size and misclassification: Is more always better?, Working paper AMSCAT-WP-97-118, AMS Center for Advanced Technologies, 1997.

Lehnert, 1993: Lehnert, Wendy G.; McCarthy, Joseph; Soderland, Stephen; Riloff, Ellen; Cardie, Claire; Peterson, Jonathan; Feng, Fang Fang; Dolan, Charles; Goldman, Seth: Umass/Hughes: Description of the CIRCUS system as used for MUC-5. In *Proceedings of 5<sup>th</sup> Message Understanding Conference*, MKP Inc., San Mateo, USA, pp. 277-291, 1993

MLArchive, 1998: Blake, Catherine L., Merz, Cristopher J.: UCI Repository of machine learning databases <http://www.ics.uci.edu/~mllearn/LRepository.html>. Irvine, CA: University of California, Department of Information and Comp. Sci., USA, 1998

Quinlan, 2000: Quinlan, Ross J.: *Data Mining Tools See5 & C5.0*, <http://www.rulequest.com/>, last visited 7/2003:

Vapnik, 1982: Vapnik, Vladimir N.: *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York, USA, 1982

Witten, 2000: Witten, Ian H.; Frank, Eibe: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2000