Formal Concepts of Learning Systems Validation in Use

Volker Döttsch*  
Universität Leipzig  
Institut für Informatik  
Augustusplatz 10–11  
04109 Leipzig  
Germany

Gunter Grieser†  
TU Darmstadt  
Fachbereich Informatik  
Alexanderstraße 10  
64283 Darmstadt  
Germany

Klaus P. Jantke‡  
Deutsches Forschungszentrum für Künstliche Intelligenz  
Stuhlsatzenhausweg 3  
66123 Saarbrücken  
Germany

Steffen Lange§  
Universität Leipzig  
Institut für Informatik  
Augustusplatz 10–11  
04109 Leipzig  
Germany

Abstract

In the problem area of evaluating complex software systems, there are two distinguished areas of research, development, and application identified by the two buzzwords validation and verification, respectively.

From the perspective adopted by the authors (cf. (O'Keefe & O'Leary 1993), e.g.), verification is usually more formally based and, thus, can be supported by formal reasoning tools like theorem provers, for instance.

The scope of verification approaches is limited by the difficulty of finding a sufficiently complete formalisation to built upon. In paramount realistic problem domains, validation seems to be more appropriate, although it is less stringent in character and, therefore, validation results are often less definite.

The aim of this paper is to exemplify a validation approach based on a clear and thoroughly formal theory. In this way, validation and verification should be brought closer to each other, for the benefit of a concerted action towards dependable software systems.

To allow for precise and sufficiently clear results, the authors have selected the application domain of algorithms and systems for learning formal languages.

Comparing the validity of one learning approach to the invalidity of another one can be seen as an interesting result in its own right.

Motivation

The increasing power of computer systems bears abundant evidence for the need of methodologies and tools for systems evaluation. Legal formalities like the German Digital Signature Act (Signaturgesetz – SigG) make the high expectations of dependability of IT systems explicit (cf. (IuKDG 1997)). The European Information Technology Security Evaluation Criteria (ITSEC 1991) and the Common Criteria for Information Technology Security Evaluation (CC 1998) specify the key requirements for meeting a large variety of security properties.

On high evaluation levels like E4 of ITSEC or EAL5 of CC, for instance, one needs to have formal models and one has to present certain reasoning steps semi-formally, at least. Verification comes into play.

In case sufficiently complete formal models and formal reasoning procedures are not at hand, one still wants to find out whether or not a given system is likely to meet the user expectations or the necessities of the application domain, accordingly. Validation is the term to denote the area dealing with these problems slightly more informally than verification does.

There are several ways in which verification and validation might come closer. One way is to attack problems together possibly dovetailing verification and validation activities, another one is to express the essentials of one approach in terms of the other.

The present paper does contribute to the latter endeavour. The authors have developed rather formal concepts of learning systems validation (cf. (Grieser, Jantke, & Lange 1997)), thus bringing validation closer to the formalisation standards of verification approaches. In the present publication, it is proved that these formal concepts work in practice. Applied to different algorithms resp. systems developed for formal language learning, these concepts allow for discriminating valid systems from those being invalid. Validation meets verification.

The Application Domain

The learning algorithms resp. systems being subject to validation are supposed to learn formal languages.

A minimal collection of necessary formalisms will be
introduced almost informally. (Gold 1967) is the seminal paper underlying our learning paradigm invoked and (Angluin & Smith 1983) is an excellent survey in this respect.

The target class of formal languages to be learnt is specified via some concept of acceptors: containment decision lists. The learning theoretic investigation in (Sakakibara & Siromoney 1992) has drawn our attention to this quite simple type of decision lists. Informally speaking, a containment decision list (CDL, for short) is a finite sequence of labelled words \((w_i, d_i)\) \((i = 1, \ldots, n)\), where the labels \(d_i\) are either 0 or 1 and the final word \(w_n\) equals the empty word \(e\). Such a list can be easily understood as a classifier for words over the underlying alphabet \(\Sigma\) as follows. Any word \(w\) fed into a CDL is checked at node \((w_1, d_1)\) first. If any check reveals that \(w_i\) is a subword of \(w\), the input word is classified as determined by \(d_i\). For instance, \(w\) is accepted exactly if \(d_i = 1\). If otherwise \(w\) does not contain \(w_i\), the input word \(w\) is passed to check it against \(w_{i+1}\). By definition, \(w_n\), which equals \(e\) is a subword of \(w\), and thus the process described terminates in a well-defined manner.

The CDL \(L = [(aab, 1), (aa, 0), (e, 1)]\) is an illustrative example. For simplicity, we choose the alphabet \(\Sigma = \{a, b\}\). Roughly speaking, the language accepted by \(L\) is the set of all words containing \(aab\) or not containing a square of \(a\). The complement consists of words containing \(aa\), but not containing \(aab\). Subsequently, we identify a CDL with the language which it accepts. What is actually meant will become clear from the context.

There are many ways to present information about formal languages to be learnt. The basic approaches are defined via the key concepts test and informant, respectively. A text is just any sequence of words exhausting the target language. An informant is any sequence of words labelled alternatively either by 1 or 0 such that all the words labelled by 1 form a text whereas the remaining words labelled by 0 form a text of the complement of the target language.

Already the fundamental results in (Gold 1967) imply that arbitrary containment decision lists are known to be learnable from informant. In other words, the knowledge contained in any CDL can potentially be acquired by processing only finitely many labelled words of the language accepted by it. More formally speaking, there is a learning algorithm that, when successively fed any informant for any CDL, outputs a sequence of hypotheses which stabilizes on a correct hypothesis for the target CDL.

In (Aha, Kibler, & Albert 1991), there has been presented some simple and alluring case-based learning algorithm named IB2. It is designed for acquiring knowledge like CDLs from finitely many cases. IB2 is selectively collecting cases which are subsequently presented. For our purpose, we have to extend IB2 to IB2' which allows for an adaptation of the similarity concepts in use. (There is definitely no hope for IB2 to learn any non-trivial CDL.) In (Sakakibara & Siromoney 1992), a particular learning algorithm, subsequently called FIND, has been proposed. FIND learns CDLs even in the presence of noise. The learning algorithms IB2' and FIND are totally defined, i.e. they respond to all possible input data.

In the present paper, we validate these learning algorithms, thereby adopting the theoretical concepts of learning systems validation developed in (Grieser, Jantke, & Lange 1997). The series of supporting experiments have been performed using the validation toolkit TIC (cf. (Burghardt, Dötsch, & Frind 1996)) which was especially designed and implemented for evaluation of several inductive learning algorithms.

**Validation Scenarios**

The focus of the present paper is on interactive approaches to the validation of complex interactive systems as exemplified in (Abel, Knauf, & Gonzalez 1996; Abel & Gonzalez 1997; Knauf et al. 1997), and others. Inspired by (Turing 1950), KNAUF is advocating in (Abel, Knauf, & Gonzalez 1996) and in his subsequent publications a type of approaches to systems validation by systematic interrogation. For specific classes of target systems, KNAUF's perspective has been adopted and adapted in (Arnold & Jantke 1997; Grieser, Jantke, & Lange 1998a; Jantke 1997), for instance.

Based on these ideas, in (Grieser, Jantke, & Lange 1998b) the authors developed and investigated a formal scenario for validating inductive learning systems. Validation by experimental interrogation is performed through the essential stages of test case generation, experimentation, evaluation, and assessment. Test cases are generated in dependence on some intended target behaviour and, possibly, with respect to peculiarities of the system under validation. Interactive validation is performed by feeding test data into the system and - hopefully - receiving system's response. The results of such an experimentation are subject to evaluation. The ultimate validity assessment is synthesized upon the totality of evaluation outcomes. In the following, we adapt this scenario for the validation of systems that are designed to learn containment decision lists.

In case a given system clearly employs some basic principle, we even face the validation of learning principles (cf. (Beick & Jantke 1999)).
A validation problem for an inductive learning system of the type under consideration is given as a triple of:

- a set \( U \) of containment decision lists,
- some learning algorithm \( S \), and
- a criterion of success.

The precise question is whether \( S \) is able to learn all CDLs from \( U \) with respect to the considered criterion of success.

In the validation community, there is an ongoing debate about the necessity of system knowledge. The question is to what extent black box validation without processing any extra system knowledge may be successful. In particular, this is very unlikely for learning systems validation (cf. (Jantke & Herrmann 1999)). Fortunately, in our study, we can always assume that the learning algorithms under inspection meet some particularly useful properties.

**Test Data Generation**

Test case generation is a quite mature research and development area in its own right. There is a large variety of approaches ranging from those with a mostly mathematical background (cf. (Auziśi et al. 1991), e.g.) to those which are exclusively founded in the humanities (cf. (Schaad 1993; Stager 1993), e.g.). The key questions are what characterizes suitable test cases, how do they relate to systems under inspection, and how to derive them.

In the present application domain, the most general approach is the following: Let \( L \) be a CDL in \( U \). Test data for this CDL are pairs of a word and the classification which \( L \) assigns to it. \( TD(L) = \{(w, c) \mid w \in \Sigma^*, c = L(w)\} \) is the set of all test data for \( L \). Test cases consist of a certain CDL \( L \) and a set of test data for it. \( TC = \{(L, E) \mid L \in U, E \subseteq TD(L)\} \) is the set of all potential test cases for CDLs in \( U \).

In practice, we have to deal with two problems. First, the set \( U \) which the learning algorithm is supposed to learn may be infinite. However, one can test the learning algorithm's ability for finitely many CDLs, only. An initial approach is to select them rather randomly. Another idea is to select "typical" CDLs of increasing complexity. In our experiments, we select 2 typical CDLs by hand. Second, the set of test data for each individual CDL is infinite. But clearly, one can probe the learning algorithms on a finite fraction of this data set, only. To be fair, it should be guaranteed that the finite fraction selected is, at least in principle, sufficiently large to learn the target CDL from it. In our special case, bounds for the size of such sufficiently large test data sets are known (cf. (Dötsch & Jantke 1996; Sakakibara & Siromoney 1992)). In our experiments, we deal with test data sets of size 300 to 8000. This is clearly above the theoretical bounds.

Once test data have been generated, they can be taken as a basis for experimentation.

**Experimentation**

Experimentation is a key activity in scientific discovery, in general (cf. (Popper 1959)). Even if the goal of investigations is as narrow as the validity of artifacts of a certain type, systematic experimentation is deemed essential (cf. (Gonzales 1997), e.g.).

Scenarios of experimentation determine how to interact with a certain target system. The learning algorithms which are subject to validation process finite lists of test data instead of finite sets of it. Moreover, one has to be aware that the output of the algorithms may depend on the order as well as the frequency in which the test data occur. This complicates the experimentation slightly. On the other hand, we exclusively deal with learning algorithms that generate an output on every possible input which, in contrast to (Grieser, Jantke, & Lange 1999b), yields some simplification.

The experimentation has a particular parameter, namely a procedure which, given any test case \( (L, E) \in TC \) and two intensity parameter \( n \) and \( t \), determines (maybe even non-deterministically) a finite collection of \( n \) lists of length \( t \) on which the target learning algorithm is tested. As a rule, each of these lists contains only test data from \( E \), possibly with repetitions. In our experiments, these lists are created by drawing randomly and independently 2000 test data from \( E \). Moreover, we choose the intensity parameters \( n = 100 \) and \( t = 2000 \). Now, let \( j \leq 2000 \) and let \( H_j \) be the multi-set of the hypotheses which the learning algorithm outputs after processing all initial segments of length \( j \). The triple \( (L, E, H_j) \) is called a protocol.

**Evaluation**

Protocols about individual experiments are subject to the expert's evaluation marked by a number between 0 and 1, expressing the opinion whether or not the experiment witnesses the learning algorithm's ability to learn the target CDL. This realizes a (partial) mapping \( Eval \) which is said to be an expert's evaluation function. Then, a tuple \( (P, b) \) consisting of a protocol \( P = (L, E, H) \) and the corresponding expert's evaluation \( b = Eval(L, E, H) \) is said to be a report.

Naturally, the question arose whether or not one may invoke computer programs to evaluate the protocols which the experimentation yields. As the results in (Grieser, Jantke, & Lange 1999b) impressively show, the evaluation of learning algorithms resp. systems require a degree of expertise which, in general, computer programs cannot achieve. Consequently, hu-
man experts have usually to be involved to rate the
goodness of the protocols. Fortunately, the learning do-
main and the learning algorithms under consideration
posses convenient properties which allow us to imple-
ment a sufficiently meaningful evaluation. In fact, this
is a corollary of Proposition 29 in (Grieser, Jantke,
& Lange 1998b). Since the language which a CDL defines
is a regular one, it can uniformly be decided whether
or not the actual hypothesis of the learning algorithm
IB2 resp. FIND describes the target CDL correctly.
However, to get deeper insights, in our study we use
additionally the following evaluation function: For a
fixed length m and all h ∈ H, let q(h) be the ratio of
all words in Σ∗ of length less or equal m that h cor-
rectly classifies. Then, Eval(L, E, H) is the fraction of
∑ all h ∈ H q(h) and the cardinality of H. In our experi-
ments, we set m = 12.

The evaluated protocols establish elementary build-
ing blocks for validity assessment.

Assessment

For complex systems, it is a huge step from evaluating
local features to an overall validity assessment. Some
authors suggest numerical approaches towards combing
sets or sequences of reports (cf. (Knauf & Gonzalez
1997), e.g.), which result from interactive experimen-
tation, into monolithic validity assessments. Others
(like (Terano 1994), e.g.) prefer a structural combina-

The CDL L1 defines an acceptoz for words over the
terminial alphabet Σ1 = \{a, b\}, whereas the CDL L2 is
used as a classifier for words over the larger terminal
alphabet Σ2 = \{a, b, c\}.

For probing the learning algorithms in case that the
CDL L1 should be learnt, we fix a test data set E1
which consists of all test data of the form (w, L1(w)),
where w is a word over Σ1 of length less or equal 7.
For our experiments, we select randomly 100 lists of
length 2000 out of the elements in E1 and create proto-
cols which reflect the algorithms' behaviour when pro-
cessing, for every of these 100 lists, initial segments
of length 10, 50, 100, 500, 1000, and 2000, respec-
tively. Figure 1 displays the essentials of the resulting
validation dialogue, i.e. it shows how these protocols
are rated by the evaluation specified in Subsection 3.3.
The variance is indicated by vertical intervals.

Additionally, Figure 2 displays the essentials of the
resulting validation dialogue for the case that the

Figure 1

Validatiom Results

In our case study, we perform a couple of experi-
ments in order to validate the learning algorithms IB2'
and FIND. In particular, we probe these learning al-

For our experiments with the validation toolkit TIC
(cf. (Dötsch & Jantke 2000)), we select two test CDLs
of increasing complexity, namely

\[ L_1 = \{(abb, 0), (abb, 1), (ab, 0), (bbaa, 1),
(aa, 0), (ba, 1), (e, 0)\} \]

\[ L_2 = \{(bc, 1), (aab, 0), (acac, 0), (ec, 1),
(ab, 0), (aa, 0), (a, 1), (b, 1), (e, 0)\}. \]

The variance is indicated by vertical intervals.
CDL $L_2$ defines the target of the learning task. In contrast, now the 100 randomly selected lists of length 2000 are formed from the elements in a test data set $E_2$ which consists of all test data $(w, L_2(w))$, where $w$ is a word over $\Sigma_2$ of length less or equal to 8.

Looking at the curves depicted, one is tempted to conjecture that the learning algorithm FIND is doing a pretty good job, i.e. it learns the target CDLs. And indeed, a closer analysis of the corresponding protocols proves that, after processing the complete lists of length 2000, FIND outputs hypotheses that are equivalent to the target CDLs $L_1$ and $L_2$, respectively. Hence, since FIND is a conservative learning algorithm, it is proved that FIND learns both CDLs on every informant that has one of the processed lists of length 2000 as initial segment.

On the other hand, the curves depicted for the learning algorithm IB2' show that the accuracy of its hypotheses is remarkably below 100%. Again, as a closer look in the protocols exhibits, IB2' never outputs a correct hypothesis. Hence, IB2' performs at least not as good as FIND. Moreover, taking into account that IB2' is unable to learn in situations in which FIND performs well, one may conjecture that IB2' is rather invalid, while FIND seems to be valid.

Interestingly, the validation statements derived nicely fit with insights that a theoretical analysis of the learning algorithms IB2' and FIND discovered. (Sakakibara & Siromoney 1992) formally proved that FIND learns every CDL on every informant for it, whereas (Dötsch & Jantke 1996) showed that there are CDLs and informants for them such that IB2' is unable to learn these CDLs on the corresponding informants.

Conclusions

In dependence on the degree of precision and completeness of the knowledge available, the assessment of an IT system’s quality can be based on more or less formal methods and tools. A high degree of precision and tool support is desirable and, sometimes, even required.

The area where systems’ behaviour is evaluated against informal requirements is named validation.

In previous publications, the authors have developed a formal framework of validation concepts. These concepts are applied. They are implemented using the validation toolkit TIC. Validity assessment is based on a certain visualization of sequences of validation reports.

The concepts have successfully been used for proving a learning system’s validity as well as for clearly demonstrating a learning system’s invalidity.

Dovetailing the validation scenario with knowledge about system properties has lead to definite validation results. The authors are convinced that the future lies in a sophisticated integration of validation and verification.

References


