Machine Learning Techniques for Civil Engineering Problems

Yoram Reich
Department of Solid Mechanics, Materials, and Structures,
Faculty of Engineering, Tel Aviv University, Ramat Aviv 69978, Israel
Phone: +972-3-640-7385, Fax: Phone: +972-3-640-7617
email: yoram@eng.tau.ac.il

November 10, 1996

To appear in *Microcomputers in Civil Engineering*, special issue on Machine Learning

Abstract

The growing volume of information databases presents opportunities for advanced data analysis techniques from machine learning (ML) research. Practical applications of ML are very different from theoretical or empirical studies, involving organizational and human aspects, and various other constraints. Despite the importance of applied ML, little has been discussed in the general ML literature on this topic. In order to remedy this situation, we studied practical applications of ML and developed a proposal for a seven-steps process that can guide practical applications of ML in engineering. The process is illustrated by relevant applications of ML in civil engineering. This illustration shows that the potential of ML has only begun to be explored, but also cautions that in order to be successful, the application process must carefully address the issues related to the seven-step process.

1 Introduction

Over the last several decades we have witnessed an explosion in information generation related to all aspects of life including all engineering disciplines. There has been an increase in active information collection to be used for solving critical engineering problems such as infrastructure management [89]. One notable example of data collection is the National Bridge Inventory in the US. In most information collection cases, information has been accumulated without knowing how it will be analyzed or used, and to date, no major practical benefit has been gained from these data collection endeavors.

Recently, a new set of techniques for knowledge extraction from data has emerged from machine learning (ML), which is a branch of artificial intelligence (AI). The original objective of ML techniques was the automated generation of knowledge for its incorporation in expert systems. This generation was expected to alleviate the knowledge acquisition bottleneck often associated with the construction of expert systems. While there have been demonstrations of knowledge acquired by single ML techniques (e.g., [52]), there has not been significant practical progress in using single ML techniques as regular tools by engineers due mainly to two reasons. First, practical problems are often too complex to be handled by a single method and second, the task of applying ML techniques in engineering practice is much more complex than described in those early studies; it is not simply a matter of taking a program and applying it to data.

To overcome the limitations of existing learning techniques with respect to the first reason, ML researchers postulated that the solution to diversity and complexity in learning situations requires the use of multiple ML techniques. Such *multistrategy learning* [54] would enable the variety of information available for learning to be taken into account.

In general, two levels can be identified within the multistrategy approach to learning [70]: the *macro* and the *micro*. The macro level deals with the use of a collection of learning programs, each addressing a separate learning problem even though they interact. It is the *non-trivial* task of the user to assemble these techniques and resolve their interactions. The micro level deals with the development of new learning programs that employ a variety of fine-grained learning strategies for solving a specific learning task.

An example of a multistrategy learning program is BRIDGER, an experimental system developed to explore the extent to which ML can aid in the creation of design support systems [68, 79]. At the macro level, BRIDGER's learning task was manually decomposed into two subtasks, learning synthesis knowledge and learning redesign knowledge, with a predefined interaction scheme. Each of these tasks was assigned to a different learning program: ECOBWEB and EPROTOS (these are enhancements of COBWEB [27] and PROTOS [8], that among other improvements, can handle continuous-valued attributes). At the micro level, each of these programs used several learning strategies to accomplish its subtask. Other examples of multistrategy systems are MOBAL (micro and macro, [59]), MLT (macro, [42]), and MCS (micro, [13]).

ML techniques can be viewed not only as knowledge generation tools but more generally, as data analysis or information modeling tools similar to traditional statistical techniques. Both statistical and ML techniques can be viewed as approximating functions. Nevertheless, ML (and some recent statistical) techniques are non-parametric, making fewer assumptions about the data, at the expense of additional computations which became possible due to the increase in the power of computers.

One example of using ML techniques for modeling involved the modeling of a decision procedure (DP) for selecting among mathematical models that simulate ground water contaminant transport processes [81]. The modeling employed two ML programs: CN2 [21] and IND [16]. Training examples for these programs were generated by simulating the DP. The programs created different models of the DP that led to its better understanding, which in turn, led to the detection of errors and to the subsequent improvement of the DP. The new DP was then subjected to the same modeling procedure.

The aforementioned projects, one for the knowledge acquisition role and one for the information modeling role, suggest that solutions to practical ML problems require the use of multiple ML methods for providing different and complementary functionalities and perspectives. A successful application requires matching the scope of applicability of ML tools to the nature of the application. This matching requires an intimate understanding of ML techniques and being creative in their operation.

But there is more to successful applied ML than using multiple ML techniques; the whole application process needs to studied. This paper addresses this issue in the context of civil engineering. The state of applied ML in civil engineering is reviewed (section 2) and the status of preliminary guidelines for using ML techniques is summarized (section 3). A seven-stage process, called Contextualized ML Modeling (CMLM), is developed from an earlier version [67] (section 4). The utility of this process needs to be tested and its future improvement depends on using it in developing practical ML applications. The stages are explained using the studies reviewed and other ML sources. Section 5 summarizes the key points and presents some future work.

2 State of the art of using ML in civil engineering

The first uses of ML programs in civil engineering involved testing different existing tools on simple problems (e.g., [6, 50, 74, 86]), gradually, more difficult problems were addressed (e.g., [32, 65, 96]), and recently, the solution of few complex practical problems have been explored (e.g., planning wastewater treatment plant operation [83], and architectural design [9]).

In many early studies, as well as many contemporary, a single ML technique has been employed. By and large, the selection of these techniques was based on availability and not necessarily applicability of the ML technique to the target problem. Often, the problem representation used was a simplification driven by the limitation of the available ML technique. There have been exceptions to this practice. In some cases, new techniques or modifications of existing techniques were developed to expand the applicability of ML techniques (e.g., for learning synthesis knowledge in Bridger [78], for architectural design in FABEL [9], or for monitoring water treatment plants [43, 83]). In other cases, several methods and creative knowledge representations were used to address different variations of learning problems (e.g., modeling material stress-strain relations [32]).

While addressing increasingly complex problems, the necessity to integrate several ML techniques for solving them was recognized and an initial theoretical foundation for such integration was developed [75]. Several subsequent systems that dealt with large problems employed multiple methods (e.g., Bridger, FABEL, [83]). These systems also incorporated new or significantly adapted ML tools.

The role assigned to ML techniques in civil engineering applications varied significantly. There have been studies on knowledge extraction (e.g., feasibility of wind bracing [6] and environmental impact assessment [38]); studies solving complete problems in which learning played a major role (e.g., cable-stayed bridge design [68] and monitoring water treatment plants [83, 43]); and studies that employed learning as part of their operation (e.g., steel bridge design [1], highway truck load monitoring [29], transmission line towers design [60], and architectural design [9]).

In addition, there have been studies directed at information modeling for creating estimation models (e.g., material stress-strain relations [32] and properties of composite materials [15]) and studies on modeling aimed at improving the understanding of a phenomenon (e.g., decision procedures [81] and material corrosion behavior [82]). The latter studies employed multiple ML techniques.

There are two issues that put work on ML in civil engineering into perspective. First, studies to date on ML applications in civil engineering have explored a small number of ML techniques, most notably supervised concept learning with few exceptions employing unsupervised learning (e.g., BRIDGER) or other techniques. This is in contrast to the potential that many other ML techniques offer [76]. Thus, the use of ML in civil engineering is only at its infancy. Second, many previous studies contained little or no systematic testing and have had little or no follow-up work. This suggests that many of these studies were preliminary and did not mature. It also cautions us to critically review the conclusions of these studies.

3 Preliminary guidelines for using ML techniques

ML techniques and problems can be characterized along many dimensions that influence the applicability of the techniques and thus should be consulted when selecting between them. These dimensions include: (1) the complexity of representing input data and learned knowledge; (2) the mechanisms for learning knowledge (or models) and the functional form of these models; (3) the mode of learning whether batch or incremental; (4) the amount of background knowledge employed while learning; (5) the ability to handle missing values, noisy, or numeric data; (6) the computational complexity of the algorithm; and (7) the learner user interaction.

In addition to these dimensions we stress two additional dimensions because they create clusters that somewhat correspond to three classical approaches to ML (see Figure 1). These dimensions are: *comprehensibility* and *timing*. The *comprehensibility* dimension ranges from (1) *clear*—in systems that create clear, comprehensible knowledge (e.g., rule induction); to (2) *opaque*—for example, in black-box systems such as neural networks. The *timing* dimension ranges from (1) *early* in systems that learn pro-actively by receiving data, learning and storing knowledge that can subsequently be used for problem solving; to (2) *late* in systems that learn reactively by storing data and subsequently retrieve it, learn from it (locally), and adapt it to solve problems. Note that the task of storing the data may involve learning to index the data efficiently.

Most ML systems learn pro-actively except for instance-based learning (IBL) systems (e.g., [97]) or case-based reasoning (CBR) systems (e.g., [9], [43], or [83]) that are reactive learners. On the comprehensibility dimension, IBL or CBR are placed in the middle, somewhat closer to symbolic learning because while their description might be clear, it might contain irrelevant data and the knowledge embedded in them might not be so clear. Decision trees might be clear or difficult to understand depending on their size and nature of data.

The comprehensibility and timing dimensions create three clusters of ML approaches shown in Figure 1. Most studies on ML in civil engineering can be classified into one of these clusters. Particular systems will be placed along these dimensions depending on the specific mechanisms that they employ. For example, ECOBWEB mainly learns pro-actively by creating a hierarchical classification of designs; this hierarchy is less comprehensible than rules or decision trees.

From the previous paragraphs we see the many dimensions influencing the behavior of ML techniques. To be successful, the application of ML to engineering practice requires careful and systematic analysis that identifies appropriate ML techniques for solving the learning tasks we wish to perform. This requirement has begun to be addressed in various studies

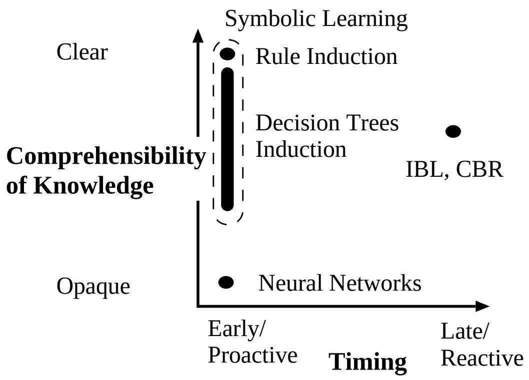

Figure 1: Key dimensions of learning systems

including:

- Developing a method for the contextual use of ML for solving complex engineering problems [67, 68, 73].
- An analysis of the difficulty and importance of various stages in the process of applying ML in engineering and a proposal to support these difficulties [80].
- Developing guidelines for the selection of ML techniques to match the characteristics of learning problems [41]. (We borrow some details from this paper in the present study.)
- An analysis of the process, issues, and steps involved in applying classification techniques in practice [14]. (We borrow some details from this paper in the present study.)
- Developing guidelines for using back-propagation neural networks in applications [35].
- The development of a system that selects the best model among three available for learning classification knowledge in a divide-and-conquer strategy [13].
- A comparison between several ML programs recommending an order for using them on classification problems [91].
- A study comparing between 23 ML classification programs on 22 datasets to derive various guidelines for their practical use [55]. This comparison is significantly different from the one in the previous item or from other comparisons in the literature. It included testing many state-of-the-art programs, sometimes several from each type; the comparison was carried out on many databases rather than one or several, thus reducing a potential bias in favor of some programs.
- A characterization of classification programs to permit selecting among them for a given learning task properties [30]. The process is driven by meta-learning from previous test results including those from the previous item.
- The development of a consultant expert system to aid practitioners in using the machine learning toolbox (MLT) [42] which includes 10 ML programs integrated together [85]. This system is based on knowledge acquired from ML users and experts. The system evolved in a longitudinal study through several versions used in practice.

In civil engineering, there have been other studies that briefly mentioned a sequence of steps for applying ML techniques (e.g., [45] or [94]). In spite of all the above studies, the complexity of applying ML techniques to practical problems is not well appreciated or understood. The following section discusses the process of applying ML to practical problems in detail, using examples from studies on applying ML in civil engineering.

4 Using ML programs: Contextualized ML modeling

Solutions to many problems follow several steps leading from problem analysis to solution deployment (see Figure 2). Some of these steps (e.g., 1 and 2, or 2, 3, and 4) may be executed in parallel or even in reverse order and the process may iterate before a successful and acceptable solution is obtained. As in many sequential procedures (even if iterative), the initial steps are the most influential on the success of the overall process but the least understood, structured, or appreciated. There are many constraints that may be imposed on this process including: time, availability of tools or information, or a required solution quality. To each practical problem, there may be several candidate solutions satisfying the requirements and the constraints. The application of ML to solving practical engineering problems follows similar steps and share similar characteristics.

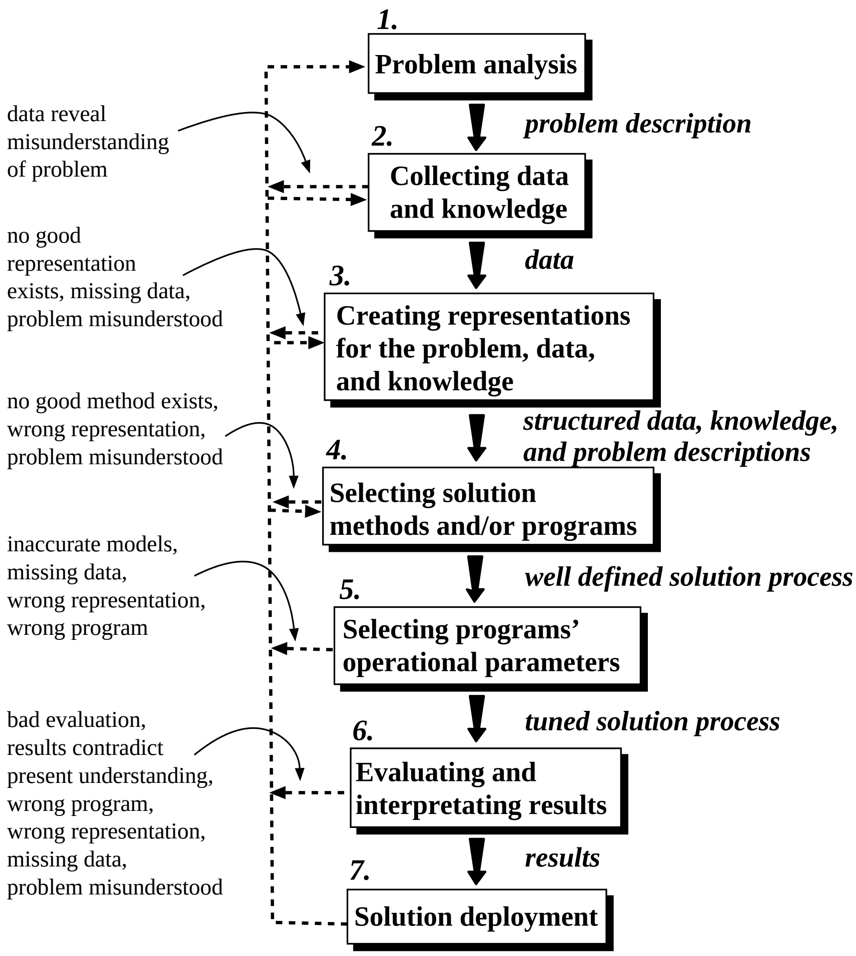

Figure 2: A model of engineering problem solving

Practical experience in solving problems using ML techniques and knowledge about the properties of these techniques can uncover characteristics of problems and their mapping to suitable ML techniques. Such a mapping can be used to select and apply ML techniques in a routine fashion. We already mentioned several studies directed at creating such a mapping (e.g., [30], [41], and [85]), but in most cases, a straight forward selection and application will not suffice. Problems will be simplified to match the capabilities of ML techniques (e.g., in BRIDGER as well as in most other studies), solution methods will be adapted (e.g., in BRIDGER) or newly developed (e.g., [38], [43], and [83]), and their use may therefore be termed as innovative or creative.

The following subsections detail seven steps that systematically address the critical issues involved in building ML applications. These steps together is a proposed procedure expected to lead to the development of successful ML applications.

4.1 Problem analysis

Engineering problems are hard to formulate and what needs to be learned may often be unclear or poorly understood. Although good formulations are very critical to the success of learning, little has been done to address the problem analysis step in the context of using ML or other data analysis tools [14, 80, 85]. When addressing real world engineering problems, the learning task cannot be formulated precisely at the beginning of a project since the task must be based on deep understanding of and involvement in this project. Thus, as shown in Figure 2, problem analysis may have to be executed iteratively and continually. These observations also hold in research projects [68], but it is rare that the evolution of ideas in such projects is reviewed. Included in the problem analysis are the following aspects:

- *The data and domain knowledge availability.* Good data and domain knowledge are key to successful application of ML. Often data is unavailable or is in insufficient quantity. Problem analysis should identify the status of data. The access to domain experts is critically important to successful ML applications. Domain experts are needed for assisting developers in better understanding the problem, in data preparation, and in result interpretation and evaluation.
- *Time constraints.* This aspect influences the availability of experts and data and the number of iterations through the steps in Figure 2 possible in a particular project.
- *The human and organizational aspects.* This aspect is critically important because if potential users are not involved in the application development, the proposed solution will probably fail even if preliminary studies show significant potential [71]. Successful real world applications of ML were developed collaboratively by ML researchers and users (e.g., SKICAT [25], MLT).
- *The status of a project.* Part of the problem analysis deals with classifying the status of the ML application. There can be several status classifications: (1) Proving of or illustrating concepts (e.g., [6], [34], [82], [86], and [96]); this requires a focused effort to yield convincing results that point to a new research or development direction. (2) Conducting preliminary studies (e.g., BRIDGER, [15], [29], [32], [38], and [93]) that deal with several steps but do not attempt to build a practical system. (3) Developing practical systems (e.g., [83]); these projects must follow the seven steps including users feedback.

  The status of a project determines the focus on some aspects of the problem. Nevertheless, it is important not to make choices in proof-of-concept or preliminary studies that might prevent them from scaling to deal with the real problems from which they emerged.

- *The goal of learning.* This goal can be automated knowledge creation or modeling in general. For each of these types, there can be several variations. A diagnosis problem will require learning predictive knowledge (e.g., [6], [38], [94], and [95]); an evaluation or estimation problem may require learning a continuous-valued function (e.g., [15], [29], [32], and [93]); and a synthesis problem will need generative knowledge (e.g., [37] and [77, 79]). While the previous problems require the use of the most applicable ML technique, modeling problems that focus on understanding (e.g., [81] and [82]) can benefit from the use of diverse techniques.

The importance of problem analysis cannot be over-emphasized. Careful problem analysis can lead to: addressing complex problems (e.g., prove the concept of learning design knowledge [66]); isolating parts of complex problems that can be solved by ML (e.g., preliminary study of improving finite element mesh [48]); and identifying sets of practical problems that can benefit from ML (e.g., modeling problems [81], damage detection [7, 88, 93], and estimation problems [15, 29, 32]).

In order to benefit from the collective experience of problem analysis conducted in many studies, it is critical that over time, more longitudinal studies are performed that document the evolution in problem understanding and proposed solutions. Beside [68], this topic has been neglected in civil engineering research.

4.2 Collecting data and knowledge

This stage is intertwined with problem analysis. Often the data improves the problem understanding and often no reasonable understanding is possible until data has been collected and studied. Data can be collected from various sources including:

- *Experts* can provide domain knowledge (e.g., [51] and [58]), case data or description language (e.g., [6], [78], [83], and [94]), and can evaluate the results of learning (e.g., [6], [22], [58], [81], and [94]). Experts can also assist in interactive learning (e.g., [6], [58], and [86]).
- *Historical records* include data available or scattered in old records or in the literature (e.g., bridge design data [78, 79], design failure data [86]) as well as more recent repositories of collected information (e.g., plant operation data [83], or accident records [5, 97]). Such data is often incomplete or inhomogeneous, requiring pre-processing or even the development of new ML techniques to handle it.
- *Published experimental data* include tabulated data, usually created for analysis purposes other than ML (e.g., material data [32, 82]). Similar to historical record data, this data is often incomplete. When such data are collected from multiple sources it require special treatment [82].
- *Simulations* can be used to generate data when ML techniques are used to solve inverse problems such as (e.g., material modeling [15], error estimation of finite element analysis [96], debugging finite element input files [95], signal plan generation [84], bridge loading identification [29], and damage detection [7, 88, 93]). Simulations can also be used when ML is used to model analytical or other tools (e.g., decision procedures [81], or expert systems [20]). Data created by simulating existing systems may leave many

missing values in the data; these values require special treatment [81].

Several important issues influence the data and knowledge collection [41, 80]. We will illustrate those related to learning classification models. Large datasets will often lead to better covering of the domain thus result in learning more honest models. However, in classification, not only is the size of the dataset important but also are the distribution of data across the classes. Failing to obtain well distributed data or failing to treat such data appropriately may lead to large errors [5, 63]. This has been termed the problem of small disjuncts [63]. Available data must be representative of the domain and should contain enough contextual information (e.g., include all environmental conditions in empirical material data [82]).

In civil engineering applications, the size of data ranged from several examples in proof-of-concept studies to several thousands when data were created by simulations. Data from historical records ranged from several tens to several hundreds. The number of attributes ranged from less than ten to 70-80. The number of classes ranged from two to 18. In one study they were 17 classes and the problem of small disjuncts was manifested [5]. If negative examples are unavailable, closed world assumption can sometimes be used to generate them [22].

One critical and time consuming issue is the integration of *multiple sources* of information and the processes integrating them, e.g., selecting the sources, reconciling their terminology, and selecting the information that can best aid learning [80]. Creating terminology for use in data and knowledge descriptions is a critical and non-trivial task [18]. In order that ML users turn past cases into useful sources of information, these cases have to be described meaningfully relative to the present problem and the knowledge associated with it. Even when this is done, descriptions are always from the point of view of those recording them and tend to incorporate only a partial understanding or partial information of the overall problem [82]. This may require iterating to complete the data, and expert assistance in data pre-processing. Few studies have recognized the terminology problem and designed the learning approach to handle such diversity by processing sub-terms [49], by interactively creating high level terms [86], or by planning to use natural language processing to create terminological structures for data pre-processing [80].

As shown in Figure 2 Step 2, the data collection process may reveal that the problem analysis was imperfect thus, requiring iterations through the problem analysis stage.

4.3 Creating representations for the problem, data, and knowledge

Once cases are collected from various sources, a schema for representing them must be devised. The schema determines much of the learning bias which significantly impacts learning. Learning bias defines the search space for models that could fit the data. Incorrect bias leaves the models that best explain the data or the more "natural" knowledge structure outside the search boundaries. Through the collection of experience in applied ML and intimate problem understanding, recent ML research has attempted to gather heuristics for selecting appropriate learning bias [33].

The dimensionality of data as reflected by the number of attributes and classes highly influences the ability to learn. Pre-processing of the data can reduce its dimensionality.

In one study, a complex real domain described by 38 attributes was transformed into a representation of 11 attributes which included numeric attributes discretized by experts and few other discrete attributes [83]. Preliminary results of this study suggest that this pre-processing captured sufficient information for learning. In another study, data described by 20 original attributes was transformed into a description of 12 attributes where all the numeric attributes have been discretized [5]. The large error rates (i.e., 63-78%) suggest that this pre-processing might have lost information that otherwise could have been useful for learning, that the original data was insufficient, or that the ML program employed was not the best for the task.

Pre-processing can also change the "nature" of the data. Dimensional analysis can lead to learning from information that is less dependent on the particular data measurements but that captures more of the "internal structure" of the data and may lead to better models [34]. Normalization of data that otherwise employ different scales can make the data more "homogeneous" and improve the ability to learn from it [29]. Sometimes, domain knowledge (and not dimensional or other analyzes) can suggest that the ratios between attributes provide additional information that is useful for learning thus leading to the construction of new ratio attributes [69, 79, 94]. In cases when the number of attributes representing data is large, feature extraction can extract attributes more meaningful for learning (e.g., using techniques borrowed from computer vision [96]).

The choice of attributes is very important for learning. Attributes create an a-dimensional space, where a is the number of attributes describing examples. For IBL or CBR, determining a metric on this space is critical to operation. Euclidean metric is not always the optimal one. Neural networks can be viewed as creating a metric on this space [62]. A wrong choice of the type of attributes can contribute to poor performance (e.g., if using continuous instead of binary attributes [51]). Also, the discretization of continuous attributes and class values [96] may result in coarse models or introduce noise due to imperfect discretization boundaries.

Good representations are key to problem solving in general and certainly for learning. Problems that do not seem to be easily solvable can become such if clever representations are devised that capture the essence of the data with minimal schema complexity (e.g., description of processes [49, 86], or path dependent data [32]). In modeling path or time dependent data, if regular propositional attribute-value representation is used, the representation can become specific to the topology of the particular problem [4] and not general as it could be [32].

When learning synthesis knowledge a $n-to-m$ mapping needs to be created between n specification attributes into m design description attributes. Concept formation is suitable for this purpose because it creates one structure that captures the interaction between the attributes [77]. In learning bridge synthesis knowledge, concept formation has been used [69, 79]. In another study on learning bridge synthesis knowledge, m separate $k-to-1$ mappings were created, where k ranged from n to $n+m-1$ [20]. The first $n-to-1$ mapping predicted one design description attribute which was in turn use as input in the next $n+1-to-1$ mapping. This scheme introduced ordering on the design decisions but also could capture some interaction between the design description attributes.

So far we mentioned propositional attribute-value representations. However, this is not a good representation for many problems that require relational or first-order logic (FOL) representations. For example, bridges are better described in a structured representation, however, in studies on learning bridge synthesis knowledge (e.g., BRIDGER, [20]) the problem was simplified into propositional attribute-value representation. This transformation lost some information. Very few civil engineering learning problems were actually represented with relational (e.g., [47], [86]) or FOL representations (e.g., [22]).

As shown in Figure 2 Step 3, if no good representation can be found the process must re-iterate. The reasons underlying the failure might be that the problem is misunderstood, that the data and domain knowledge are insufficient, or that the problem does not fit available representations in which case it needs to be coerced to fit the most applicable representation.

4.4 Selecting solution methods and/or ML programs

The selection of solution methods or learning programs should be based on the properties of the learning problem formulated in previous steps. We have already mentioned studies that attempted to give guidelines for such selection through conducting cross-validation (CV) testing of different techniques [91] (see further in subsection 4.6); by meta-learning from previous comparative testing [30, 55]; by building an expert system [85]; or by describing selection guidelines [41]. In subsection 3 we mentioned characteristics of learning systems that determine their capabilities and applicability. We now elaborate on them.

- *The complexity of representing input data and learned knowledge.* This dimension ranges from propositional attribute-value representation in most ML applications, structured or relational in some applications (e.g., [47], [49], and [86]), to first-order logic (e.g., [22]). Rules, decision trees, hierarchical classifications, or memories of IBL are all propositional attribute-value representations. Memories of cases in CBR can be complex (e.g., relational in FABEL), but not always are they manipulated by the learning mechanisms.

  Complex representations allow to handle complex domains more naturally, and might be able to handle background domain knowledge better. In contrast, complex representations complicate the learning process.
- *The mechanisms for learning knowledge (or models) and the functional form of these models.* There are many such mechanisms including various greedy search techniques as in recursive partitioning tree building, beam search as in CN2 or AQ, or hill climbing as in ECOBWEB. The type of search determines the size of problem that can be addressed by the approach and the ability to approximate the globally optimal model.

  Viewed as mappings, learned models can be linear or non-linear; they can map a description of an entity onto one attribute as in classification, or map n input attributes onto m output attributes as would be required in synthesis [77].
- *The mode of learning whether batch or incremental.* When dealing with small datasets this dimension is unimportant. However, when dealing with large practical datasets, and especially those that evolve with time, it is important to learn incrementally. All rule or decision tree learning programs used in civil engineering have been batch learners, whereas CBR or IBL are naturally incremental. Neural networks can also learn incrementally although this has not been used in civil engineering applications.
- *The amount of background knowledge employed while learning.* In most ML systems background knowledge is implicit in the description language. Some systems allow to incorporate constraints on the learned models such as required attributes (e.g., IILS

[38]), or precedence between attributes (e.g., CN2 as in [81]) when learning rules. Logical inference systems allow the incorporation of diverse knowledge (e.g., [11, 22]). Domain knowledge can constrain the search of the learning system. In addition, if learned knowledge is used as background knowledge, the learning system can be used to mimic an incremental mode.

- *The ability to handle missing values, noisy, or numeric data.* In all the representations, the type of attributes, whether input or output can be discrete, ordered, structured, or numeric. It is better to handle different attribute types without modifications to prevent from losing information. Indeed there are many techniques that handle such data. However, in situations where systems are selected based on availability those abilities may be lacking, requiring using default for missing values, discretizing numeric attributes, or ignoring noise. Such modifications to data may deteriorate learning performance from the level that might have been attained by learning from the original data.

  Another issue is the representation of the "class" attribute when learning classification models. In many cases this attribute is originally numeric and cannot be handled by many ML techniques and thus is been discretized (e.g., [96]). Example techniques that can handle numeric class attributes are IBL or CBR techniques (e.g., [40] or Ecobweb), regression trees (e.g., CART [12]), few decision tree induction programs (e.g., NewID [10]), and some types of neural networks.
- *The computational complexity of the algorithm.* This dimension is tied to the mode dimension. For batch learners the complexity issue is more acute since any addition of data requires re-starting the learning process. Approaches vary considerably along this dimension, often in relation to the complexity of representing input data and knowledge. Below are the computational complexities of learning from a dataset, of various techniques [3, 21, 27]:
  - $O(a^2n)$ for tree induction with discrete attributes (e.g., ASSISTANT [19]);
  - approximately $O(a^2n \log n)$ for tree induction with some numeric attributes (e.g., C4.5 [64], IND, NewID);
  - $O(ab^2n \log_b n)$ for Ecobweb, but each incremental step costs only $O(ab^2 \log_b n)$;
  - roughly $O(a^2sn^2)$ in CN2 or AQ-type [53] algorithms;
  - $O(anm)$ for IBL, but each incremental step costs only $O(am)$; and
  - exponential in the number of states for inducing relational grammar [47];

  where, a is the number of attributes describing examples, n is the number of examples, s is the maximum star size for CN2 or AQ, b is the branching factor of Ecobweb's classification hierarchy (usually 3 or 4), and m is the number of examples retained in IBL memory (which can be as large as n).

  For many ML systems, especially those newly developed in civil engineering research, the complexity figure is not given and it is unclear whether these systems can scale to handle real domains.
- *The learner user interaction.* In most ML systems there is little interaction between the user and the program, although, such interaction is critical when addressing real problems. It is often necessary to manually guide the search of the learning system or input new data to guide it (e.g., [6], [86]). Systems that do provide such interaction often originate from knowledge acquisition research but also from ML research (e.g.,

Protos [8]/Eprotos, MLT, IND).

In most applications, ML programs are selected because of availability and not because they are actually the best for the particular task with respect to the aforementioned dimensions [68]. Often, the choice between one of the three clusters of approaches in Figure 1 is based on researchers inclination and not applicability. In such cases, the representation "supposedly natural" to the domain, is selected to fit the learning program of choice.

Even if programs are selected based on steps 1 to 3, it is rare that the reasons underlying this selection are well articulated or that their validity is reviewed by testing the learning system after its deployment. Of course there are exceptions to the above practice (e.g., [38], [43], [9], [83], and [68]). One particular example discusses the process of building a learning system (i.e., Bridger) by going through the evolution of selecting programs, evaluating them, and finally developing an approach to handle the complexities discovered in previous tests. The cycles through problem analysis and system testing improved the understanding of the learning task. Such a record of longitudinal study is necessary to appreciate the program selection step and its relation to problem analysis.

When modeling data with ML this step may be less critical. Instead of creating the best possible model, it is more important to create different models that can provide different perspective of the data [81].

As shown in Figure 2 Step 4, if no good method exists, the process needs to re-iterate. The reasons for this failure might be that the problem is misunderstood, that the wrong representation was selected, or that there is no available ML technique that is applicable to the problem.

4.5 Selecting program operational parameters or options

The use of most ML programs involves setting up various parameters or selecting between various options. Different parameters give rise to different performance levels of these programs. In a recent study comparing 23 ML programs on 22 datasets, programs default parameters were used [55]. Other studies selected tuning parameters according to general recommendations and attempted to confirm them (e.g., [7] according to [61]). However, these examples are exceptions to a common practice among researchers who tune operational parameters to obtain the best possible performance of their learning program on their demonstration problems. For neural networks, parameters selection is more critical since some parameters determine the topology of the network while others tune the performance. These selections, together with selecting the learning rule and the activation function, are aimed at leading to good network convergence and generalizability of results.

In general, the selection of parameters do not generalize to new learning contexts. The reason underlying this unfortunate problem is that different parameters control different aspects of programs behavior (e.g., the complexity of the schema, the handling of noisy data, or the amount of searching allowed) and it is not easily discerned which ones contribute to good or bad performance in a particular learning context. In consequence, the *a priori* selection of parameters that will result in good performance in a particular learning context remains non-trivial.

Some programs such as IND are suites of many techniques whose behavior can be configured by selecting different options and not only operational parameters. Selecting among these options is as difficult as the setting of operational parameters. Over the last few years there have been an increase in the number of studies dealing with selecting between different options. In particular, studies on decision tree induction explored the selection between tree pruning strategies [56] and the selection between attribute splitting rules (e.g., [57], [17], [46]).

A better appreciation of the difficulty of these issues can be gained by following the evolving understanding of tree splitting rules in its relevant references, showing a reversal in the conclusions derived in those studies. The cause of the reversal was the methodologically wrong use of the option selection procedure in the first study (i.e., ref. [57]) [17, 46]. The inability to verify the origin of the first conclusion was due to insufficient data about the testing procedure employed in that study. This example demonstrates that great care should be exercised in experimental testing of programs if these tests are expected to lead to some operational conclusions.

Similar to the selection between different programs, the importance of selecting program parameters or options is different depending on the role assigned to ML techniques [81]. In any case, an appropriate selection of operational parameters can be done using various testing procedures that are discussed in the next section. It is hoped that gradually, experience in using ML tools in practice would lead to formulating guidelines for the *a priori* selection of operational parameters [85].

The selection of operational parameters is the first step that employs quantitative evaluation of learned models. If the goal of learning is automated knowledge generation, this evaluation is critical. If no parameter combination leads to good performance of the ML program selected in the previous step, there is a mismatch between the data, its representation, and the ML program. As shown in Figure 2 Step 5, the process can re-iterate by selecting another program, revising the representation, or collecting additional data and domain knowledge. If all these fail, the problem might be misunderstood or beyond the capabilities of existing ML techniques.

If the goal of learning is modeling for the purpose of understanding, the critical evaluation is deferred to the next step. In this case, better understanding can sometimes be gained even by using imprecise or inaccurate models [81, 87].

4.6 Testing, evaluating, and interpreting results

This stage is the least attended to by the research community although it is the foundation of all scientific and practical work. To illustrate, consider a recent study that detailed minimum requirements for evaluating neural networks [28]. One interesting part in this study was an examination of publications in two leading neural networks journals discovering the poor evaluation practice they exercised. The status of testing in civil engineering applications of ML is similar and sometimes worse. In some studies, no report on testing is mentioned. Other studies provide one anecdotal example showing what the technique might be doing in a particular situation. Even studies that perform some testing may be deficient if their tests are methodologically wrong, or if they do not complete the testing procedure, leaving readers to interpret the results.

The testing of ML techniques involves assessing the quality of knowledge or models they create. This assessment is inherently multi-dimensional and includes quantitative and qualitative aspects such as:

(1) quantitative estimation of model accuracy;
(2) quantitative comparison between the predictive accuracy of one ML technique and some base-line performance which may be: another ML technique, other computational tools such as expert systems (e.g., [39]; although the statistical test in this reference is incorrect), default rules, or expert judgment;
(3) qualitative experts' interpretation of learned knowledge or models (e.g., [20] and [81]) or qualitative comparison with expert generated rules (e.g., [94]);
(4) qualitative improvement in understanding the data used to train the ML technique by experts or users of the learned models (e.g., [81]); and
(5) practical benefits of the deployed system.

We now elaborate on this list.

Quantitative estimation of model accuracy. There are several methods that have been used to estimate the performance of ML techniques. They are summarized in Table 1. In the Table, the *number of internal iterations* denotes the number of times a basic procedure is executed in order to obtain one estimation, $\hat{\theta}$, of the true accuracy θ. The *number of iterations* denotes the number of times the complete process is performed. The two iteration figures influence the computational cost of the method, although for special cases such as decision trees, there are ways to reduce this cost by dynamically creating only the path in the tree that is needed in each testing [46].

Table 1: Properties of performance estimation methods

Estimation method	Size of training set	Size of testing set	Number of internal iterations	Number of iterations	Method variability	Method bias
Resubstitution	n	n	1	1	very high	very optimistic
Hold-out	70-80% n	20-30% n	1	I (O(10))	high	pessimistic
Cross-validation	(k-1)/k n	1/k n	k (∼10)	1	high	nearly unbiased
Bootstrap	see text	see text	see text	I (∼200)	low	slightly optimistic

The *method variability* reflects the estimated error of the method when applied to different datasets drawn from the same distribution of the present dataset. Given a sample of I *independent* estimations, $\hat{\theta}_i$, $i = 1 \ldots I$, each calculated in one iteration, this variability is calculated by

$$\sqrt{\frac{\sum_{i=1}^{I}(\hat{\theta}_i - \bar{\hat{\theta}})^2}{(I-1)I}},$$

where $\bar{\hat{\theta}}$ is the average of the estimations. However, as we see later, when using any of the methods, the I estimations are not independent thus the above formula is incorrect and estimating the variability is not trivial.

The *estimation bias* reflects the difference between the expected value of the estimation, $E(\hat{\theta})$, and the parameter being estimated, θ, i.e., $bias = E(\hat{\theta}) - \theta$. The bias is important when estimating the absolute accuracy of a ML technique. It is less critical when comparing between two techniques since both can be assessed by the same biased method. For such comparison, method variability is more critical.

The common methods used to estimate the performance of ML techniques are:

- Resubstitution: In this method, all the examples in the dataset are used for training as well as for testing the model. This method produces a very optimistic upper-bound estimation of accuracy, i.e., its error estimation is biased downward. Assuming that the dataset is sampled from a large population, the performance of resubstitution is highly dependent on this sampling, i.e., it has high variability.

- Hold-out: In this method, the data is randomly divided into a training (about 70-80% of the examples) and a testing set (remaining 20-30%). In order to produce results with a confidence interval of about 95%, the testing set should include more than 1000 examples; otherwise, this method may produce poor results. In smaller datasets, this method is often repeated several tenth of times, but the results have high variability that is dependent upon the initial random subdivision, in addition to the variability due to the sampling of the dataset from the larger population. Note that these repetitions are not independent, having used the same dataset. The results of this method may be pessimistic because not all available data is used for training. The majority of studies in civil engineering have used these inferior performance estimation techniques even when the testing data were small.

- k-fold cross-validation (CV) or leave-one-out: In order to remedy the problems in the hold-out method, a different method for using the data for training and testing is employed. Figure 3 illustrates a k-fold CV method. It has been common in general ML studies to use a 10-fold CV method when the number of examples, n, exceeds 100, or a leave-one-out method (i.e., $k = n$) for small datasets [36, 55, 92]. Several civil engineering studies have also used CV [4, 5, 72, 79, 81, 82]. In order to obtain good results, a stratified CV method should be used [26] in which each subset contains examples with the same class distribution as in the complete set. Similar practice can be used in the hold-out method. CV has high variability with respect to the sampling of the dataset. Furthermore, its estimation is significantly dependent on the subdivision into subsets.

- Bootstrap: This estimation method reduces the variability observed in previous methods and are only slightly optimistic [23]. In the bootstrap, a sample of n examples is drawn with replacement from the original n examples. On an average, $1-1/e = 0.63$ of the original examples are drawn into this sample. The new sample is used for training and the old sample for testing. The result of this testing provides a measure of the optimism of resubstitution. I such samples are drawn and their optimism measure is calculated. The final estimation is the average of these measures added to the resubstitution estimation.

In cases where program parameters are optimized for some database, there is a need to test the accuracy of the method on an independent test set that was not used to tune the parameters in order to prevent obtaining too optimistic estimations. Figure 4 illustrates such

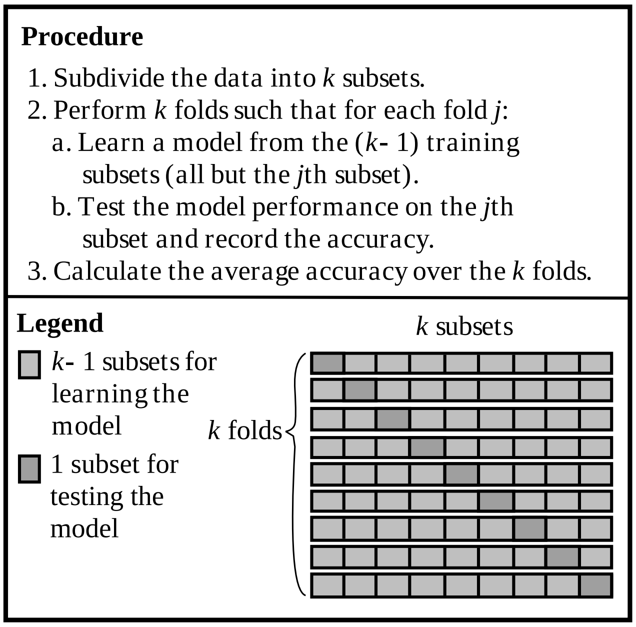

Figure 3: Performance accuracy estimation using a k-fold cross-validation method

an estimation procedure that can be used for tuning operational parameters and options of programs, and finally, estimating the performance accuracy of the learned knowledge. In the first step, the data is subdivided into data for model learning and model testing. In the second step, the data for model learning is used to select the best model (i.e., learning approach) and operational parameters. In the third step a model is created from the complete model learning set by the best approach and best operational parameters. This model is tested on the testing set. Obviously, the final experiment is a hold-out estimation method with its above limitations.

Quantitative comparison between the predictive accuracy of one ML technique and some base-line. In order to conduct such comparison it is useful to develop sets of benchmark problems. The general ML community collects such problems in a database repository. One example of such a simple set was used in [90], [34], and subsequently in [2] to compare the performance of different neural networks approaches and illustrate some modeling issues. Each study replicated earlier results to get an appropriate base-line for its version of the previous approaches: an important activity that is rarely practiced. Note that such replication is easily done when dealing with neural networks since it is easy to reconstruct and test them. Also note that the particular set of problems was easy to solve and thus their use as benchmark problems might not reveal the “true” comparative behavior of the approaches on larger scale problems.

It has been common to compare between different programs by calculating their performance accuracy using CV without performing any statistical analysis. This procedure may lead to wrong conclusions due to the high variability of CV. Similarly to executing a hold-out method several times one should consider doing the same for CV with different subdivisions. This can control the variability due to subdivision but not the variability due to the sampling of the original dataset. Since different iterations of hold-out or even internal iterations

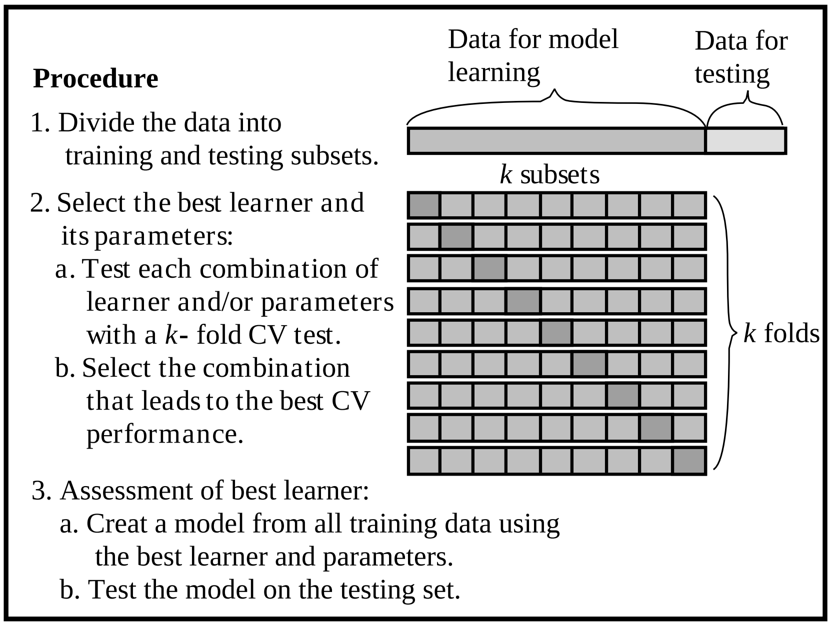

Figure 4: Three-stage performance accuracy estimation

of CV use almost the same data, they are not independent. Therefore, common statistical tests for small samples such as the standard t-test, for testing whether the difference between the means of two independent distributions is statistically significant, have to be modified to take care of this dependence [26]. One modification is the use of a paired sample t-test. This requires that all programs use the same training and testing data in parallel. It poses a problem because one cannot use published results that do not specify how to replicate the data subdivisions for the purpose of employing this statistical test.

Moreover, it is even inappropriate to use this modified test when several ML techniques are comparatively tested on several databases [26] because there is a chance that some of those many comparisons will yield unwarranted results. Thus, if no correction is performed to common tests to account for this chance, their conclusions may be wrong. This problem has been termed the *multiplicity effect* and it can even be manifested during the internal operation of ML programs (e.g., when determining which rules generated in a rule learning program are significant [31]).

Qualitative evaluation of learned knowledge. This evaluation is subjective and has to be performed with care. It is critical to exercise multiple comparisons or evaluate models by several experts to get an inter-subjective opinion. If an evaluation involves comparing learned models with other rules or models, a particular set-up should be followed in which one group of experts evaluates the learned models with respect to a base-line and another group — serving as the control group — evaluates another arbitrary set of rules (i.e., "placebo" rules) compared to the same base-line. The experts must not know to which of the groups they belong. Furthermore, the models and base-line rules need to be presented to experts in a balanced manner: some experts will see the learned models first and others the base-line rules first. It is outside the scope of this paper to discuss statistical experimental design in depth; nevertheless, it is important to understand the necessity of carefully exercising qualitative evaluations of models. Without such care, the results might be anecdotal.

Assessment of the practical benefits of the deployed system. There is no example of an application of ML in civil engineering that proved successful in practice. In such assessment, as in assessing the introduction of any new technology, care should be exercised to make sure that the benefits observed do indeed follow from the deployed system and not from the "excitement" with the new technology. Often such excitement leads to temporarily paying additional attention to some work aspects thus displaying apparent benefits. Such improvements, however, fade away quickly.

As shown in Figure 2 Step 6, if the evaluation results are negative, the process needs to re-iterate. The reasons for a failure can be problem misunderstanding, wrong selection of program, wrong use of representation or missing data and domain knowledge.

4.7 Solution deployment

There are virtually no practical solutions developed using ML for real civil engineering problems. However, some experience with fielded applications of ML exist elsewhere and also the assimilation in civil engineering of computer systems in general provides insight on this issue [71].

As far as the technical details are concerned, it turns out that many general practical applications of ML used the simplest and most robust ML techniques, namely, tree or rule induction programs. The product of the application ranged from manual use of learned models for improving understanding or for manual execution (e.g., [24] and [81]) to the embedding of the learned models in large software systems (e.g., SKICAT) [44]. The particular technical details are influenced by the many issues that were outlined in the problem analysis step, in particular, the human and organizational aspects. These aspects mandate that an infrastructure is built around the technical product that will address training users, preparing usable manuals, and building mechanisms for maintaining the product until it matures and later on, during its life-cycle.

In some sense, this stage constitutes the real testing of an application. It validates that the technical solution, be it learned models or software are used to the advantage of users, and that the solution deployed addresses the original problem as formulated in the first step and as may have evolved during development. A practical application can fail due to any of the above mentioned issues and other unnoticed factors. Hence, all issues beyond the technical aspects need to be given utmost attention. Any failure, can trigger iterating through the seven-step process.

5 Discussion and Summary

Building practical applications of ML requires competence in dealing with the many issues discussed in this paper. They can be summarized in several key points:

- Better understanding of the nature of different learning problems is critical and can be improved by studying previous applications and trying to form characterizations of engineering domains [30]. Any characterization must be built to evolve continually. This requires establishing a repository of ML programs and civil engineering datasets or problem definitions similar to the repository established by the general ML community.

- Careful task analyzes and clever problem formulations can transform difficult to manageable problems (e.g., [48], [66], and [83]).
- Intimate understanding of ML techniques can be used to map problem characteristics into ML techniques that can address them. This requires that researchers continually update themselves with state-of-the-art ML research.
- In order to perform comparative studies or when the learning role is modeling, it should be easy to employ several ML techniques simultaneously on learning problems. This can be facilitated by building toolboxes (e.g., MLT and IND). In neural networks applications this has been practiced due to the relative ease of implementation or the availability of neural network tools in commercial software such as MATLAB. Comparative studies can lead to improving ML techniques (e.g., as in the StatLog project [55]).
- We elaborated on the issue of evaluation due to its poor status yet critical importance. Any evaluation, whether it involves a single or several ML techniques, and whether it is quantitative or qualitative, requires careful attention in its design, analysis, and interpretation. Without such attention, conclusions may be anecdotal, caused by random factors, or simply incorrect.
- In order to improve the understanding of the seven-step CMLM process and evolving it, longitudinal contextual studies must be performed and documented. The documentation must include the difficulties and failures encountered and not only the successes. Gradually this data will lead to creating rules for the "routine" use of ML techniques and later will provide some insight about "innovative" or "creative" uses.
- In order to succeed in applying ML in practice there is a need for an information management system to support the application process [73, 80]. This system must support balanced interaction between users and ML programs and other utility functions. Such an interaction ability has been realized in systems such as MOBAL, MLT, or IND to a small extent.

Most of the studies with ML techniques in civil engineering have employed supervised concept learning tools. The same situation has been observed in relation to practical applications of ML in general [44]. Given the general potential of these techniques, we have barely started to use them in solving problems. The many studies referenced in this paper point to many opportunities.

We believe that these opportunities could be materialize into practical systems if the application process is carefully executed following the seven-step CMLM process. We also intend to continuously update this process using feedback from future ML applications.

References

[1] Adeli, H., and Balasubramanyam, K. V. A novel approach to expert systems for design of large structures. *AI Magazine 9*, 4 (1988), 54–63.

[2] Adeli, H., and Park, H. S. Counterpropagation neural networks in structural engineering. *Journal of Structural Engineering 121*, 8 (1995).

[3] Aha, D. W., Kibler, D., and Albert, M. K. Instance-based learning algorithms. *Machine Learning 6*, 1 (1991), 37–66.

[4] Arciszewski, T., Khasnabis, S., Hoda, S. K., and Ziarko, W. Machine learning in transportation engineering: A feasibility study. *Applied Artificial Intelligence 8*, 1 (1994), 109–124.

[5] Arciszewski, T., Michalski, R. S., and Dybala, T. STAR methodology-based learning about construction accidents and their prevention. *Automation in Construction 4*, 1 (1995), 75–85.

[6] Arciszewski, T., Mustafa, M., and Ziarko, W. A methodology of design knowledge acquisition for use in learning expert systems. *International Journal of Man-Machine Studies 27*, 1 (1987), 23–32.

[7] Barai, S. V., and Pandey, P. C. Performance of generalized delta rule in artificial neural networks in damage detection. *Engineering Applications in Artificial Intelligence 8*, 2 (1995), 211–221.

[8] Bareiss, R. *Exemplar-Based Knowledge Acquisition.* Academic Press, Boston, MA, 1989.

[9] Börner, K., Ed. *Modules for Design Support.* Technical Report FABEL-Report No. 35. GMD, Sankt Augustin, Germany, 1995.

[10] Boswell, R. Manual for NewID version 4.1. Tech. Rep. TI/P2154/RAB/4/2.3, The Turing Institute, Glasgow, 1990.

[11] Bratko, I., and Muggleton, S. Applications of inductive logic programming. *Communications of the ACM 38*, 11 (1995), 65–70.

[12] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees.* Belmont, Waldsworth, CA, 1984.

[13] Brodely, C. E. Addressing the selective superiority problem: Automatic algorithm.model class selection. In *Proceedings of the Tenth International Conference on Machine Learning* (1993), Morgan Kaufmann, pp. 17–24.

[14] Brodley, C. E., and Smyth, P. Applying classification algorithms in practice. *Statistics and Computing (in press)* (1996).

[15] Brown, D. A., Murthy, P. L. N., and Berke, L. Computational simulation of composite ply micromechanics using artificial neural networks. *Microcomputers in Civil Engineering 6*, 2 (1991), 87–97.

[16] Buntine, W., and Caruana, R. *Introduction to IND Version 2.1 and Recursive Partitioning.* NASA Ames Research Center, Moffet Field, CA, 1992.

[17] Buntine, W., and Niblett, T. A further comparison of splitting rules for decision tree induction. *Machine Learning 8*, 1 (1992), 75–86.

[18] Buntine, W., and Stirling, D. Interactive induction. Tech. Rep. TIRM-88-030, The Turing Institute, Glasgow, 1988.

[19] Cestnik, K., and Bratko. Assistant-86: A knowledge-elicitation tool for siohisticated users. In *Progress in Machine Learning* (Bled, Yugoslavia, 1987), I. Bratko and N. Lavrač, Eds., Sigma Press, pp. 31–45.

[20] Chen, Q., and Arciszewski, T. Machine learning of bridge design rules: A case study. In *Proceedings of the 2nd Congress on Computing in Civil Engineering* (New York, NY, 1995), ASCE, pp. 711–718.

[21] Clark, T., and Niblett, P. The CN2 induction algorithm. *Machine Learning 3*, 4 (1989), 261–283.

[22] Dolšak, B., and Muggleton, S. The application of inductive logic programming to finite-element mesh design. In *Inductive Logic Programming* (London, UK, 1992), S. Muggleton, Ed., Academic Press, pp. 453–472.

[23] Efron, B. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the Americal Statistical Association 78*, 382 (1983), 316–331.

[24] Evans, B., and Fisher, D. Overcoming process delays with decision-tree induction. *IEEE Expert 9* (1994), 60–66.

[25] Fayyad, U. M., Smyth, P., Weir, N., and Djorgovski, S. Automated analysis and exploration of large image databases. *Journal of Intelligent Information Systems 4* (1995), 7–25.

[26] Feelders, A., and Verkooijen, W. Which method learns most from the data? In *Preliminary papers of the Fifth International Workshop on Artificial Intelligence and Statistics* (1995), pp. 219–225.

[27] Fisher, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine Learning 2*, 7 (1987), 139–172.

[28] Flexer, A. Statistical evaluation of neural network experiments: minimum requirements and current practice. In *Proceedings of the Thirteenth European Meeting on Cybernetics and Systems Research* (1995), R. Trappl, Ed.

[29] Gagarin, N., Flood, I., and Albrecht, P. Computing truck attributes with artificial neural networks. *Journal of Computing in Civil Engineering 8*, 2 (1994), 179–200.

[30] Gama, J., and Brazdil, J. Characterization of classification algorithms. In *Progress in Artificial Intelligence, 7th Portuguese Conference on Artificial Intelligence, EPIA-95* (Berlin, 1995), C. Pinto-Ferreira and N. Mamede, Eds., Springer Verlag, pp. 189–200.

[31] Gascuel, P., and Caraux, G. Statistical significance in inductive learning. In *Proceedings of the 10th European Conference on Artificial Intelligence* (New York, NY, 1992), B. Neumann, Ed., Wiley, pp. 435–439.

[32] Ghaboussi, J., Garrett, J., and Wu, X. Knowledge-based modeling of material behavior with neural networks. *Journal of Engineering Mechanics 117*, 1 (1991), 12–153.

[33] Gordon, D. F., and DesJardins, M. Evaluation and selection of biases in machine learning. *Machine Learning 20*, 1-2 (1995), 5–22.

[34] Gunaratnam, D. J., and Gero, J. S. Effect of representation on the performance of neural networks in structural applications. *Microcomputers in Civil Engineering 9*, 2 (1994), 97–108.

[35] Hegazy, T., Fazio, P., and Moselhi, O. Developing practical neural network applications using back-propagation. *Microcomputers in Civil Engineering 9*, 2 (1994), 145–159.

[36] Henry, R. J. Methods of comparison. In *Machine Learning, Neural and Statistical Classification* (Chichester, England, 1994), D. Michie, D. Spiegelhalter, and C. C. Taylor, Eds., Ellis Horwood Publishers, pp. 107–124.

[37] Ivezic, N., and Garrett, J. A neural network-based machine learning approach for supporting synthesis. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8*, 2 (1994), 143–161.

[38] Julien, B., Fenves, S. J., and Small, M. J. Knowledge acquisition methods for environmental evaluation. *AI Applications 6*, 1 (1992), 1–20.

[39] Kamarthi, S. V., Sanvido, V. E., and Kumara, R. T. Neuroform—neural network system for vertical formwork selection. *Journal of Computing in Civil Engineering 6*, 2 (1992), 178–199.

[40] Kibler, D., Aha, D. W., and Albert, M. K. Instance-based prediction of real-valued attribures. *Computational Intelligence 5*, 2 (1989), 51–57.

[41] Kodratoff, Y., Moustakis, V., and Graner, N. Can machine learning solve my problems? *Applied Artificial Intelligence 8*, 1 (1994), 1–31.

[42] Kodratoff, Y., Sleeman, D., Uszynski, M., Cause, C., and Craw, S. Building machine learning toolbox. In *Enhancing the KNowledge Engineering Process—Contributions from ESPRIT* (Oxford, UK, 1992), B. La Pape and L. Steels, Eds., Elsevier, pp. 81–108.

[43] Krovvidy, S., and Wee, W. G. Wastewater treatment systems from case-based reasoning. *Machine Learning 10*, 3 (1993), 341–363.

[44] Langley, P., and Simon, H. A. Applications of machine learning and rule induction. *Communications of the ACM 38*, 11 (1995), 55–64.

[45] Lee, H., and Hajela, P. Vector clustering for neural network based prediction of geometrical characteristics. In *Neural Networks and Combinatorial Optimization in Civil and Structural Engineering* (Topping, B. H. V. and Khan, A. I., 1993), U. Edinburgh, Ed., Civil-Comp Press, pp. 19–29.

[46] Liu, W. Z., and White, A. P. The importance of attribute selection measures in decision tree induction. *Machine Learning 15*, 1 (1994), 25–41.

[47] Mackenzie, C. A. Inferring relational design grammars. *Environment and Planning B: Planning and Design 16*, 3 (1989), 253–287.

[48] Manevitz, L., Yousef, M., and Givoli, D. Finite element mesh generation using self-organizing neural networks. *Microcomputers in Civil Engineering (accepted for publication)* (1996).

[49] Mawdesley, M. J., and Carr, V. Artificial neural networks for construction project planning. In *Neural Networks and Combinational Optimization in Civil and Structural Engineering* (Edinburgh, UK, 1993), B. H. V. Topping and A. I. Khan, Eds., Civil-Comp Press, pp. 39–46.

[50] McLaughlin, S., and Gero, J. S. Acquiring expert knowledge from characterized designs. *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing 1*, 2 (1987), 73–87.

[51] Messner, J. I., Sanvido, V. E., and Kumara, S. R. T. StructNet: a neural network for structural system selection. *Microcomputers in Civil Engineering 9*, 2 (1994), 109–118.

[52] Michalski, R. S., and Chilausky, R. L. Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology. *International Journal of Man-Machine Studies 12*, 1 (1980), 63–87.

[53] Michalski, R. S., Mozetič, I., Hong, J., and Lavrač, N. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of AAAI-86 (Philadelphia, PA)* (San Mateo, CA, 1986), Morgan Kaufmann, pp. 1041–1045.

[54] Michalski, R. S., and Tecuci, G., Eds. *Proceedings of the First International Workshop on Multistrategy Learning.* Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1991.

[55] Michie, D., Spiegelhalter, D., and Taylor, C. C. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Publishers, Chichester, England, 1994.

[56] Mingers, J. An empirical comparison of pruning methods for decision-tree induction. *Machine Learning 4*, 2 (1989), 227–243.

[57] Mingers, J. An empirical comparison of selection measures for decision-tree induction. *Machine Learning 3*, 4 (1989), 319–342.

[58] Miyamoto, A., Kushida, M., and Kinoshita, K. Concrete bridge rating expert system with machine learning. In *Proceedings of IABSE Colloquium 1995* (1995), pp. 301–306.

[59] Morik, K., Wrobel, S., Kietz, J. U., and Emde, W. *Knowledge Acquisition and Machine Learning — Theory, Methods and Applications.* Academic Press, London, UK, 1993.

[60] Murlidharan, T. L., Aravind, H. B., Suryakumar, G. V., and Raman, N. V. Expert tower analysis and design system. II: Search strategies and learning. *Journal of Computing in Civil Engineering 5*, 2 (1991), 193–210.

[61] Pao, Y. H. *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley, Reading, MA, 1989.

[62] Pao, Y.-H., and Sobajic, D. J. Neural networks and knowledge engineering. *IEEE Transactions on Knowledge and Data Engineering 3*, 2 (1991), 185–192.

[63] Quinlan, J. R. Improved estimates for the accuracy of small disjuncts. *Machine Learning 6*, 1 (1991), 93–98.

[64] Quinlan, J. R. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1992.

[65] Reich, Y. Converging to "Ideal" design knowledge by learning. In *Proceedings of The First International Workshop on Formal Methods in Engineering Design* (Colorado Springs, Colorado, 1990), P. A. Fitzhorn, Ed., pp. 330–349.

[66] Reich, Y. Design knowledge acquisition: Task analysis and a partial implementation. *Knowledge Acquisition 3*, 3 (1991), 237–254.

[67] Reich, Y. Designing integrated learning systems for engineering design. In *Proceedings of The Eight International Workshop on Machine Learning (Evanston, IL)* (San Mateo, CA, 1991), L. Birnbaum and G. C. Collins, Eds., Morgan Kaufmann, pp. 635–639.

[68] Reich, Y. The development of Bridger: A methodological study of research on the use of machine learning in design. *Artificial Intelligence in Engineering 8*, 3 (1993), 217–231.

[69] Reich, Y. A model of aesthetic judgment in design. *Artificial Intelligence in Engineering 8*, 2 (1993), 141–153.

[70] Reich, Y. Macro and micro perspectives of multistrategy learning. In *Machine Learning: A Multistrategy Approach, Vol. IV* (San Francisco, CA, 1994), R. S. Michalski and G. Tecuci, Eds., Morgan Kaufmann, pp. 379–401.

[71] Reich, Y. What is wrong with CAE and can it be fixed. In *Preprints of Bridging the Generations: An International Workshop on the Future Directions of Computer-Aided Engineering* (Pittsburgh, PA, 1994), Department of Civil Engineering, Carnegie Mellon University.

[72] Reich, Y. Measuring the value of knowledge. *International Journal of Human-Computer Studies 42*, 1 (1995), 3–30.

[73] Reich, Y. Modeling engineering information with machine learning. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing 10*, 2 (1996), 171–174.

[74] Reich, Y., and Fenves, S. J. Floor system design in Soar: A case study of learning to learn. Tech. Rep. EDRC-12-26-88, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 1988.

[75] Reich, Y., and Fenves, S. J. Integration of generic learning tasks. Tech. Rep. EDRC 12-28-89, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 1989.

[76] Reich, Y., and Fenves, S. J. The potential of machine learning techniques for expert systems. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing 3*, 3 (1989), 175–193.

[77] Reich, Y., and Fenves, S. J. The formation and use of abstract concepts in design. In *Concept Formation: Knowledge and Experience in Unsupervised Learning* (Los Altos, CA, 1991), D. H. J. Fisher, M. J. Pazzani, and P. Langley, Eds., Morgan Kaufmann, pp. 323–353.

[78] Reich, Y., and Fenves, S. J. Inductive learning of synthesis knowledge. *International Journal of Expert Systems: Research and Applications 5*, 4 (1992), 275–297.

[79] Reich, Y., and Fenves, S. J. A system that learns to design cable-stayed bridges. *Journal of Structural Engineering, ASCE 121*, 7 (1995), 1090–1100.

[80] Reich, Y., Konda, S., Levy, S. N., Monarch, I., and Subrahmanian, E. New roles for machine learning in design. *Artificial Intelligence in Engineering 8*, 3 (1993), 165–181.

[81] Reich, Y., Medina, M., Shieh, T.-Y., and Jacobs, T. Modeling and debugging engineering decision procedures with machine learning. *Journal of Computing in Civil Engineering 10*, 2 (1996), 157–166.

[82] Reich, Y., and Travitzky, N. Machine learning of material behavior knowledge from empirical data. *Materials & Design 16*, 5 (1995), 251–259.

[83] Sànchez, M., Cortés, U., R.-Roda, I., Poch, M., and Lafuente. Learning and adaptation in wastewater treatment plants through case-based reasoning. *Microcomputers in Civil Engineering (accepted for publication)* (1996).

[84] Saraf, R., and Fisher, D. Online signal-plan generation for centralized traffic control using neural networks. *IVHS Journal (in press)* ((1995)).

[85] Sleeman, D., Rissakis, M., Craw, S., Graner, N., and Sharma, S. Consultant-2: pre- and post-processing of machine learning applications. *International Journal of Human-Computer Studies 43*, 1 (1995), 43–63.

[86] Stone, J., and Blockley, D. Towards machine learning from case histories. *Civil Engineering Systems 5*, 3 (1989), 129–135.

[87] Subrahmanian, E., Konda, S. L., Levy, S. N., Reich, Y., Westerberg, A. W., and Monarch, I. A. Equations aren't enough: Informal modeling in design. *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing 7*, 4 (1993), 257–274.

[88] Szewczyk, Z., and Hajela, P. Damage detection in structures based on feature-sensitive networks. *Journal of Computing in Civil Engineering 8*, 2 (1994), 163–178.

[89] TRR. Transportation Research Record No. 1271, Transportation Data and Information Systems: Current Applications and Needs. Transportation Research Board, National Research Council, Washington, DC, 1990.

[90] Vanluchene, D., and Sun, R. Neural networks in structural engineering. *Microcomputers in Civil Engineering 5*, 3 (1990), 207–215.

[91] Weiss, S., and Kulikowski, C. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems.* Morgan Kaufmann, San Mateo, CA, 1990.

[92] Weiss, S. M., and Kapouleas, I. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI* (San Mateo, CA, 1989), Morgan Kaufmann, pp. 781–787.

[93] Wu, X., Ghaboussi, J., and Garrett, J. Use of neural networks in detection of structural damage. *Computers & Structures 42*, 4 (1992), 649–659.

[94] Yeh, Y.-C., Kuo, Y.-H., and Hsu, D.-S. Building KBES for diagnosing PC pile with inductive learning. *Journal of Computing in Civil Engineering 7* (1992), 223–233.

[95] Yeh, Y.-C., Kuo, Y.-H., and Hsu, D.-S. Debugging finite element programs input data with machine learning. *Microcomputers in Civil Engineering 7* (1992), 223–233.

[96] Zarka, J., and Hablot, J. M. Learning expert systems in numerical analysis of structures. In *Expert Systems in Structural Safety Assessment* (Berlin, 1989), A. S. Jovanovic, K. F. Kussmaul, A. C. Lucia, and P. P. Bonissone, Eds., Springer-Verlag, pp. 305–314.

[97] Zhang, J., and Yang, J. An application of instance-based learning to highway accident frequency prediction. *Microcomputers in Civil Engineering (accepted for publication)* (1996).