Towards Practical Machine Learning Techniques

Yoram Reich[1]

In the *Proceedings of the First Congress on Computing in Civil Engineering*
Washington, DC, June 20-22, 1994, pp. 885-892

ABSTRACT

Most research on the application of machine learning to engineering problems have solved artificial problems. While research claimed to have reached results that would improve practice, these results have never been put to work by engineers themselves in solving their problems. A different approach of doing research on machine learning application is presented and a system design that may result in tangible practical results is outlined. The development of this system is underway.

INTRODUCTION

In order to make machine learning (ML) techniques usable for engineers, a methodological shift is required in the way ML research is perceived, planned, and executed. Past investigations that dealt with the development of ML techniques for solving engineering problems mostly developed ideas that were tested on simplified artificial problems. Thus, researchers could not demonstrate that their ideas had practical implications. With no direct connection between research and practice, researchers would claim that the goal of research was *understanding* the nature of learning to be formulated in some theories. Often, the explicit research goal stated that this understanding will later lead to practical results. Unfortunately, experience has shown that there is a gap between understanding or theories and practice.

To start with, theories originating in ML research, and that deal with questions such as *how many examples are required to probably learn a concept with some accuracy*, could sometime be formulated as well defined problems to be solved by formal proofs. Unfortunately, as the ML literature documents, formal ML theories have little relevance to empirical ML approaches (Angluin and Smith 1983; Turney 1991); consequently, their relevance to engineering is scant.

Not much comfort comes from other, empirical, theories such as those hypothesizing that *concept formation (i.e., incremental unsupervised concept learning) is more suitable than supervised concept learning for learning synthesis knowledge*. While these theories may have been empirically tested on artificially constructed engineering

[1]Senior Lecturer, Department of Solid Mechanics, Materials and Structures, Faculty of Engineering, Tel Aviv University, Tel Aviv 69978, Israel. Email: yoram@eng.tau.ac.il

problems and may have received some theoretical support (Reich 1993), they have not withstood serious testing with competing hypotheses. Even if they had, their practical relevance would still be questionable since they have not been tested in practical settings.

There are few recent studies that report the successful application of ML tools for solving real problems.[2] These reports, however, have not demonstrated the use of ML programs by practitioners unfamiliar with ML (or acknowledged that there may be a problem in such usage), while even users experienced with using existing ML programs on practical problems will confirm the significant difficulties in applying them even to artificial engineering problems. The latter difficulty is mainly due to the omission from research of critical issues dealing with using ML (Reich et al 1993).[3]

In order to make ML techniques practical, the nature of ML theories must shift from *understanding ML* to addressing the core of practice: *what makes ML techniques usable for engineers.* This shift involves different theorizing, and certainly different ways of doing research; in particular, research goals must be formulated and their attainment evaluated in practical settings. I will illustrate the shift by using examples from my and my colleagues' work.

Three case studies are described that span the methodological range and show the direction towards practical ML techniques: (1) the development of **Bridger**, an experimental learning system developed to explore the potential of ML for creating a support system for cable-stayed bridge design (Reich 1993); (2) the modeling of existing decision procedures with ML techniques that can facilitate their evaluation or redesign (Reich et al 1995; Shieh 1993); and (3) the learning of bridge management knowledge from bridge databases (Reich et al 1994). Based on these studies, a hypothesis is put forward about developing practical usable ML programs, that will be tested in the third project.

CASE STUDIES

Three case studies are reviewed that span a methodological range from "understanding" to "practical usability."

Bridger

Bridger is an experimental system developed to explore the possibility of building design support systems with ML techniques with no interaction with real designers (Reich 1993; Reich and Fenves 1995). The design of cable-stayed bridges was selected as a target problem that may benefit from computational support. The problem was simplified and recast into a form amenable to the exploitation of AI concepts and tools. The experimental research concentrated on the acquisition of synthesis knowledge by modifying an existing learning program, **Cobweb**, to "behave" in ways that better suited our understanding of design.

Bridger was developed using the old methodology. Its approach was theoretical and basic in that it involved studying mechanisms for learning in artificial design problems. Even though the project was justified through a practical necessity, there was no commitment to transfer the ML mechanisms directly into practice. Instead,

[2] It is interesting that the majority of these studies originated in Europe rather than in the US.

[3] This statement applies to programs such as: C4.5, ID3, FOIL, COBWEB, CN2, NewID, IND, AutoClass, CLUSTER, AQ15, Protos, PRISM, IBL, and others.

BRIDGER passed qualitative and quantitative evaluations on artificial problems. In its present form, BRIDGER may act as a source of ideas for building computational support systems for bridge designers but will not suffice as a support system for bridge engineers in any way.

Consequently, while a theoretical research perspective may claim that BRIDGER succeeded in demonstrating some ideas, a practical research perspective may contend that its development was inconsequential to practice.

Modeling Decision Procedures (MODEP)

The second project, henceforth referred to as MODEP (Modeling Decision Procedures), representing the methodological mid-range, applied existing ML techniques to model a decision procedure. The decision procedure was a program for selecting among groundwater modeling techniques developed at Duke University (Medina et al 1988) and used by North Carolina and US Air Force officials. The modeling employed the ML programs CN2 (Clark and Niblett 1989), IND (Buntine and Caruana 1992), and later FOIL (Quinlan 1990), for creating abstract representations of the original procedure in the form of rules (CN2 and FOIL) or decision trees (IND). The use of different programs' parameters created different representations (or perspectives) of the original procedure, thus facilitating its evaluation and subsequent redesign.

The project led to the following observations. (1) ML programs have limitations even for addressing tasks simpler than their intended role as automatic knowledge generation tools; these limitations constitute feedback to the developers of ML programs. Nevertheless, (2) ML programs developed in research can be applied to practical problems if the nature of the application is carefully selected to match their limited scope of applicability. This may require being intimate with the details of ML principles and programs and being creative in their operation. In contrast, (3) ML programs are hardly usable to engineers unfamiliar with such detailed knowledge of ML.

New theories were not formulated in this study; rather, it was a modest test of theories hypothesizing the practical utility of ML. The test was partially successful but, overall, it uncovered significant difficulties in the practical use of ML programs, especially for complex learning tasks. This study was instrumental in formulating the hypothesis that practical use of ML techniques requires an infrastructure for the manipulation of different ML programs in creative ways and the accumulation of information about the past uses of ML programs.

Learning from Bridge Databases

The third project deals with learning bridge management knowledge from bridge management databases for the purpose of supporting decision making (Reich et al 1994). This project adopts the practical engineering relevance as the primary goal. The theories underlying the research deal with the issues involved in making ML techniques usable in engineering practices such as: the necessity of providing flexible access to a variety of techniques or the importance of managing ML results. These theories are intended to be evaluated in a natural setting and constantly evolve with practice. There is a dual relationship between these theories and practice because (1) they are expected to aid it, (2) it improves, and therefore, (3) it drives new theorizing. We discuss the design of this project in the following sections.

THE INGREDIENTS OF A PRACTICAL ML SYSTEM

A practical ML system is an environment that facilitates the use of ML programs for practical purposes by engineers. There are several ingredients to such a system: (1) the basic building blocks that implement ML techniques; (2) an infrastructure that integrates the building blocks and additional auxiliary mechanisms needed for completing the overall engineering task; (3) a practical context that imposes real constraints and demands on the use of the techniques; (4) engineers that are expected to solve real problems with the aid of ML programs; and (5) researchers who are expected to respond to users' difficulties in utilizing the software for their tasks.

It is clear that the environment is the whole setting including the practitioners functioning in it and not just the software. This has significant ramifications for the way such an environment could be developed and evolve if only due to the fact that those expected to evolve it (i.e., researchers) are part of it. In subsequent subsection we discuss items (1), (2), and (3) and briefly touch upon items (4) and (5).

Building Blocks

The building blocks of a practical ML systems are available ML programs. To best utilize these programs the following information about each is required:

I/O — the Input/Output of the ML program, representing the learning task functionality.

KR — the knowledge representation used by the ML program, defining its scope.

C — the control structure of the program.

This information is the one required to specify *generic learning tasks* — constructs similar to generic tasks (Chandrasekaran 1986) but whose purpose is learning (Reich and Fenves 1989). It is envisioned that given this information, a ML program could be identified, adapted or developed for a particular learning task.

How does one construct the mapping between learning tasks and available ML techniques? The common response will be to use the information in the generic task description with possibly additional problem features such as: data properties (e.g., noise, number of attributes, relevance of attributes, type of attributes, percentage of missing values, etc.), or desired mode of operation of the learning task (i.e., automatic or manual). A process that is intended to guide in this mapping is $\mathbf{M^2LTD}$ (Reich 1994).

Brief experience has shown that it is hard to specify learning programs with features so as to provide enough ground for their selection when a particular learning task is requested. Rather, the selection of programs requires significant understanding of ML programs and experience in their practical usage in the particular task.

Thus, instead of trying to evolve the knowledge about the activities in $\mathbf{M^2LTD}$, such as task analysis, in response to practical feedback expressed by some problem features, we modify the process so it maintains the contextual information involved in selecting and using ML techniques and call it Contextualized ML Modeling (CMLM) (Figure 1). The management of the contextual information mandates the introduction of an integrative framework for managing the information about selecting between, applying, and interpreting results from different ML programs.

Integrative Framework

The purpose of an integrative framework is to encapsulate different ML programs with additional tools needed to solve domain problems such that ML programs could

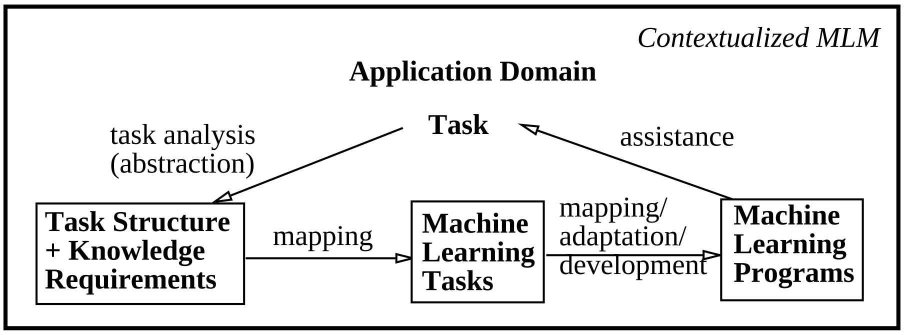

Figure 1. CMLM— Contextualized Machine Learning Modeling

be utilized by an engineer in the context of solving the actual engineering problem. Two potential types of frameworks are outlined.

The first type allows users' interaction, but enforces a particular underlying representation for capturing knowledge, examples, and concepts. One example of this type is MLT (Morik et al 1991). If such a framework provides an expressive language and mechanisms for translating information between the representations of different techniques (originally built for different representation languages), the framework could be used successfully to leverage on the strength of different programs.

The second framework type advocates for an open architecture that facilitates the integration and execution of different programs and allows specifying data exchange protocols without enforcing a single approach or representation.[4] It is understood that this flexibility may result in irreversible translations or information loses when transferring information between different programs and that some translations may require human intervention. Nevertheless, it is argued that this is a necessity since no single unified representation could support the complete learning process. One example of this integration type is n-dim (Levy et al 1993; Subrahmanian et al 1991), the framework adopted in the third project.[5] n-dim is developed to support the collaborative work of multiple designers. n-dim has facilities with which users can flexibly model various kinds of information including: task description, contextual information, data, desired output, available ML programs and their previous uses, etc.

Addressing Real Engineering Problems

When addressing real world problems, the task of ML programs cannot be formulated precisely before starting the project since the task must be based on a deep understanding of, and involvement in the project. That is, one cannot manipulate all the data with ML techniques and extract all possible knowledge structures to address any future need. Rather, the particular context of the project determines the questions whose solutions may benefit from learned knowledge.

To illustrate, consider a database containing manufacturing data including two

[4] Of course, such a capability accepts any single approach as one possible configuration. Therefore, this type subsumes the first integration type.

[5] n-dim is developed by a group that includes (in alphabetical order): R. F. Coyne, A. Dutoit, E. Gardner, S. L. Konda, S. N. Levy, I. A. Monarch, Y. Reich, E. Subrahmanian, M. Thomas, A. W. Westerberg.

fields: percent failures in quality inspection (PF) and time the product was late in delivery (TL). From the manufacturers perspective, it is beneficial to know which parameters influence PF so as to manipulate them towards its reduction. On the other hand, the buyers are interested in the TL and less in the number of the failed parts that are not within their monetary responsibility. A more detailed analysis may reveal that TL is also important to manufacturers because of contract obligations, and PF may reveal some flawed design or manufacturing planning operations that the buyer may be responsible for and thus may benefit from its reduction.

Users and Researchers

It is clear that the selection of ML programs cannot be done by using simple learning tasks' features. This selection may be difficult even when using the contextual information. Further, the necessary learning functions may be unavailable. In such situations, and due to the practical nature of using the ML program, it is critical to provide timely responses to such difficulties. Researchers are expected to provide this support or otherwise the use of ML programs in the corresponding practical setting may terminate.

AN EXPERIMENTAL SET-UP

In order to develop theories about practical ML programs and test them empirically we develop an environment with the components described in the previous section:

(1) The generic learning tasks will include several borrowed ML programs (e.g., FOIL, CN2, IND), as well as in-house developments (e.g., **Ecobweb**).
(2) The integrative framework is n-dim.
(3) The real engineering problem is the management of bridges — a difficult and critical problem. The premise of the experimental set-up is that the management of bridges could be improved by learning from records of management decisions and their consequences that are stored in bridge management databases.
(4) The users of ML programs are expected to be the engineers in charge of bridge management. It is paramount to attract them to participate in the research by becoming users of ML programs. It is only through such work that ML programs could be improved in a manner relevant to practice.

An example scenario using this set-up appears in Reich et al (1994).

It is worth observing that this experimental set-up is different from common research practice in that it will not test alternative ways of introducing ML programs into practice in any statistical sense. It would have been better if an alternative integrative framework, such as MLT would be used for comparison or if different approaches other than the one outlined would be exercised to obtain engineers' involvement. These would improve the validity of our understanding of what it takes to make ML programs practical but, unfortunately, is not feasible: No practitioner would have the resources to solve the same bridge management problem with several methods and if different methods are exercised on different problems that are contextually different, the validity of the exercise may be questioned.

Such observation does not mean that valid results could not be obtained from this approach. Contextualized research through the use of various case study methods is becoming acceptable in many social science disciplines. Making ML programs practical to engineers in a socio-technical setting requires adopting the research methods used in those disciplines.

POSTSCRIPT

The development of ML (as well as other software) techniques for practical engineering use requires a shift in the focus of theories from practice-independence to participation of researchers and practitioners, working together in the context of solving real problems.

This shift entails changing the research methodology used to allow participation of practitioners. Whether this participation is possible in practice, how it could be managed, and whether it leads to the practical improvements so desired are still open questions we hope to address in the third project.

REFERENCES

References

Angluin, D. and Smith, C. H. (1983). "Inductive inference: Theory and methods." *Computing Surveys*, 15(3):237–269.

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

Chandrasekaran, B. (1986). "Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design." *IEEE Expert*, 1(3):23–30.

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

Levy, S., Subrahmanian, E., Konda, S. L., Coyne, R. F., Westerberg, A. W., and Reich, Y. (1993). "An overview of the *n*-dim environment." Technical Report EDRC-05-65-93, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Medina, M. A., Butcher, J. B., and Marin, C. M. (1988). "An advisory system for groundwater quality modeling and management." *Hydrosoft, Computational Mechanics Publications*, 1(4):197–203.

Morik, K., Causse, K., and Boswell, R. (1991). "A common knowledge representation integrating learning tools." In Michalski, R. S. and Tecuci, G., editors, *Proceedings of the First International Workshop on Multistrategy Learning*, pages 81–96, Fairfax, VA, Center for Artificial Intelligence, George Mason University.

Quinlan, J. R. (1990). "Learning logical definitions from relations." *Machine Learning*, 5(3):239–266.

Reich, Y. and Fenves, S. J. (1989). "Integration of generic learning tasks." Technical Report EDRC 12-28-89, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Reich, Y. and Fenves, S. J. (1995). "A system that learns to design cable-stayed bridges." *Journal of Structural Engineering, ASCE.* (in press).

Reich, Y., Konda, S., Levy, S. N., Monarch, I., and Subrahmanian, E. (1993). "New roles for machine learning in design." *Artificial Intelligence in Engineering*, 8(3):165–181. Special issue on Machine Learning in Design.

Reich, Y., Fenves, S. J., and Subrahmanian, E. (1994). "Flexible extraction of practical knowledge from bridge databases." In *Proceedings of the First Congress on Computing in Civil Engineering (Washington, DC)*, pages 1014–1021, New York, NY, American Society of Civil Engineers.

Reich, Y., Medina, M., Shieh, T.-Y., and Jacobs, T. (1995). "Modeling engineering decision procedures with machine learning." *Journal of Computing in Civil Engineering.* (accepted for publication).

Reich, Y. (1993). "The development of BRIDGER: A methodological study of research on the use of machine learning in design." *Artificial Intelligence in Engineering*, 8(3):217–231. Special issue on Machine Learning in Design.

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

Shieh, T.-Y. (1993). "The potential of machine learning techniques for improving decision algorithms in groundwater contaminant transport modeling." Master's thesis, Graduate School of Duke University, Durham, NC.

Subrahmanian, E., Westerberg, A. W., and Podnar, G. (1991). "Towards a shared information environment for engineering design." In Sriram, D., Logcher, R., and Hukuda, S., editors, *Computer-Aided Cooperative Product Development, MIT-JSME Workshop (Nov., 1989)*, Berlin, Springer-Verlag.

Turney, P. (1991). "The gap between abstract and concrete results in machine learning." *Journal of Experimental and Theoretical Artificial Intelligence*, 3(3):179–190.