Machine Learning 30, 133–163 (1998)

Learning in the "Real World"

LORENZA SAITTA saitta@di.unito.it
FILIPPO NERI neri@di.unito.it
Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy

Editors: Ron Kohavi and Foster Provost

Abstract. In this paper we define and characterize the process of developing a "real-world" Machine Learning application, with its difficulties and relevant issues, distinguishing it from the popular practice of exploiting ready-to-use data sets. To this aim, we analyze and summarize the lessons learned from applying Machine Learning techniques to a variety of problems. We believe that these lessons, though primarily based on our personal experience, can be generalized to a wider range of situations and are supported by the reported experiences of other researchers.

Keywords: real-world applications, application life cycles

> "*So much research, so few good products*"
> (Isaacs & Tang, 1996)

1. Introduction

Notwithstanding the large number of Machine Learning (ML) approaches and off-the-shelf systems, significant applications with a well-documented economic and technological impact are still rare. This situation is not peculiar to ML; it applies in general to the process of *technology transfer*, as Isaacs & Tang (1996) note, especially when the source of transfer is academic research (Foley, 1996). It is difficult for new technologies to be accepted, particularly in fields where well-assessed, although limited, classical software development methods exist. According to Narendra (1996), even the connectionist approach, which has experienced a wide acceptance outside academia, produced very few reports of fielded applications in domains such as process control and industrial robotics. Documentation on the impact of ML on the industrial/economic world can be found in some review papers and books (Forsyth & Rada, 1986; Langley & Simon, 1995; Rudenström, 1995; Widrow, Rumelhart, & Lehr, 1994), in the proceedings of specialized workshops (Kodratoff, 1994b; Aha & Riddle, 1995; Engels et al., 1997), and, occasionally, in technical journals and proceedings (Asker & Maclin, 1997; Esposito et al., 1990, 1997; Giordana et al., 1993; Esposito, Malerba, & Semeraro, 1994).

A possible reason for the apparent lack of interest in large applications inside the research communities might be the feeling that technology transfer is a one-way, time-consuming process, with a limited impact on research trends. On the contrary, we strongly believe that technology transfer is inherently a two-way process, able to orient research towards the most exciting challenges. The interplay between research and application is actually a fundamental mechanism (but not the only one) driving scientific advances. In order to better understand this interplay, the application process has to be carefully analyzed to search for emergent patterns and key issues.

Motivated by the above considerations, we will discuss *what* we believe is the core of a successful ML application, and *why* people in academia should get involved in that complex and difficult process. To this end, the present paper has two related goals:

- To define and to characterize the "real-world" ML application process, to identify distinctive factors and to point out that there is much more to the application process than running an algorithm on a set of data.
- To investigate the nature and the extent of the potential feedback from applications to ML research.

In order to reach these goals, we will rely primarily on our own experience in developing and applying ML systems. Discussions with colleagues and information extracted from the literature also contributed new perspectives. We will not describe in detail either the methodologies or the systems we developed; they are documented in previous papers. Nor will we extensively report on the applications we performed, because they also have been described earlier (see, for example, Giordana et al., 1993; Faure, Frediani, & Saitta, 1993; Botta et al., 1991; Baroglio et al., 1996; Neri, Saitta, & Tiberghien, 1997).

In our research group we worked with a variety of paradigms, such as symbolic (Bergadano, Giordana, & Saitta, 1988, 1991; Bergadano & Giordana, 1988; Gemello, Mana, & Saitta, 1991; Botta & Giordana, 1993; Saitta, Botta, & Neri, 1993; Baroglio, Botta, & Saitta, 1994), genetic (Giordana & Sale, 1992; Giordana & Saitta, 1994; Giordana & Neri, 1995; Neri & Saitta, 1996a, 1996b; Anglano, Giordana, & Lo Bello, 1997; Neri, 1997), and connectionist (Blanzieri & Katenkamp, 1996; Baroglio et al., 1996; Botta, Giordana, & Piola, 1997), as well as with their integration (Giordana et al., 1997).

Moreover, we were fortunate to be given the opportunity to work on applications in different fields (using both learning approaches and, earlier, non-learning, classical AI methods) such as industrial troubleshooting (Giordana et al., 1993; Botta et al., 1991), molecular biology (Calapaj, Lo Bello, & Saitta, 1997), medicine (Belforte et al., 1985; Milanese et al., 1982), industrial robotics (Baroglio et al., 1996), speech recognition (Bergadano, Giordana, & Saitta, 1988; Faure, Frediani, & Saitta, 1993; Gemello & Saitta, 1987), cognitive psychology (Neri, Saitta, & Tiberghien, 1997) and knowledge discovery in large databases (Botta et al., 1997). Thus we have had the chance to interact with both domain experts and end-users who displayed different attitudes and different degrees of trust and understanding of the potential of the methodologies we were proposing.

The lessons that we have learned over the years can be summarized in the following claims, which we will support and elaborate upon in the rest of the paper:

- ML "real-world" applications require a user to participate actively in the application process.
- Equating the procedure of testing ML algorithms on ready-to-use data sets with a "real-world" application is a common mistake, which may have negative impact on ML research.

In order to achieve its goal, the paper is organized as follows. Section 2 contains related work, i.e., work that either presents application surveys or discusses general issues related to them. Section 3 elaborates on the first claim, providing a tentative definition of a “real-world” application and its abstract life cycle. Section 4 is related to the second claim and shows why using a ready-to-use data set cannot be considered a “real” application. The claims put forward in Sections 3 and 4 shall be supported both by considerations derived from others’ reported experiences and by the sample of our applications briefly described in Section 5. Also in Section 5, an overview of our research is provided to facilitate understanding of the content of the described applications. Section 6 presents a characterization of the different ways of performing ML applications and their potential impact on ML research. Finally, Section 7 presents some conclusions.

2. Related work

Extracting general lessons from a set of applications as diverse as those performed in the ML field is no easy task. Few papers have tried to do it. Even overview papers are scarce and scattered in a broad spectrum of publication sources.

Older surveys include those by Michie (1982), and Forsyth & Rada (1986). More recently, the papers collected by Kodratoff (1994b) provide an overview of the European ML applications performed at that time, mostly in an industrial environment. Some of the papers present broad collections of case studies whereas others concentrate on a specific application.

Langley & Simon (1995) describe six fielded applications and a dozen “realistic” ones, some of which are also reported by Rudenström (1995). Rudenström analyzes sixteen applications, four of which are fielded. He classifies the reported cases according to characteristics of the application and of the learning algorithm used, and proposed a set of features for describing an ML application.

Aha & Riddle (1995) collected eight papers on ML applications. Some of these describe concrete experiences and others extract general lessons. In particular, both Brodley & Smyth (1995) and Saitta, Giordana, & Neri (1995)[1] pinpoint key issues in the process of applying ML in practice. Brodley & Smyth (1995), in particular, present a detailed analysis of the “application,” “data,” and “human” factors relevant in the application process.

Engels et al. (1997) edited a collection including five application descriptions, and three papers by Adriaans (1997), Brodley (1997) and Langley (1997), raising general issues and stimulating discussions.

Brachman et al. (1996), Fayyad (1996), Fayyad, Haussler, & Stolorz (1996) and Fayyad, Piatetsky-Shapiro, & Smyth (1996) have presented reviews of applications and have also proposed characterizations of the application process in the recently emerging field of Data Mining and Knowledge Discovery in Databases, to which Machine Learning is a substantial contributor.

More focused overviews are presented by Bratko & Muggleton (1995), on ILP applications, by Nédellec (1995), on comprehensibility, by Michie, Spiegelhalter, & Taylor (1994), on the results of the StatLog project, and by van Someren (1994), on the relations between

Machine Learning and Knowledge Acquisition. Similar reviews exist for Neural Networks (see, for instance, Widrow, Rumelhart, & Lehr, 1994; Narendra, 1996).

Finally, Mitchell (1997) suggests potential areas in which ML is most likely to produce a significant impact.

From the above overviews a commonality emerges: as Langley (1997), Brodley & Smyth (1995) and Brodley (1997) note, most of the current applications of Machine Learning involve tasks that can be set up as supervised learning, in particular, as classification tasks. Although several of the issues we raise are more generally applicable, in the present paper we also concentrate on applications of supervised learning.

3. "Real-world" applications

In this section we define a "real-world" Machine Learning application and characterize its distinctive factors. A simple and operational definition of a "real-world" Machine Learning application is as follows.

> "A *real-world* Machine Learning application is one in which a *user* actively participates in the development process and exploits either the learned knowledge or some derivation thereof."

The above definition stems from the belief that the point of applying ML is the very result of the learning activity, i.e., either a concrete body of knowledge to be put into action to perform a task, or some piece of truly new and useful information to be exploited thereafter. Langley & Simon (1995) take the same stand. We would like to strengthen their position by observing that designers of ML systems usually envision a scenario including themselves, data and, possibly, a marginally useful expert, but almost never the user. On the contrary, the user should be a fundamental component of this scenario, because s/he establishes the "rules of the game" and the evaluation criteria, and must be eventually convinced to use the result of learning. According to the above definition, the user must not only be present, but also actively participate in the development of a "real-world" application.

The potential actors in a ML application development are the "user" (U), the domain "expert" (E) and the ML "developer" (D). As we will see in the following, all three do not need to be present in every application, nor do the terms always designate a single person. In fact, the "user" may correspond to a group of people, or even to a department of an organization, which decides about the worthiness of a product after a period of use. The same is true for the "expert" and the "ML developer." Finally, the "user" and the domain "expert" may actually correspond to the same person or group of persons. So, by U, E, and D, we intend to denote "roles" rather than actual people.

The degree of interaction among the three actors ranges between two extreme situations, which we shall call "stand-alone" and "collaborative," respectively. These two modes of operation, synthesized into two life cycles, will be discussed in the next subsections. Brodley & Smyth (1995), Langley & Simon (1995), Schwabacher, Hirsh, & Ellman (1995), Asker & Boström (1995), Langley (1997), and Létourneau, Famili, & Matwin (1997) propose schemes analogous to, but more abstract than, the ones described in this paper; their schemes do not allow the more subtle distinctions we want to make.

3.1. "Collaborative" process of application development

We begin with the collaborative scheme (shown in figure 1) because many considerations are also valid for the stand-alone scheme. The most obvious feature of figure 1 is its iterative nature, which is commonly acknowledged, for example, by Brodley & Smyth (1995), Asker & Boström (1995), Schwabacher, Hirsh, & Ellman (1995), Brodley (1997), and Asker & Maclin (1997). In fact, at each point of the cycle, one may be required to go back to any previous one.

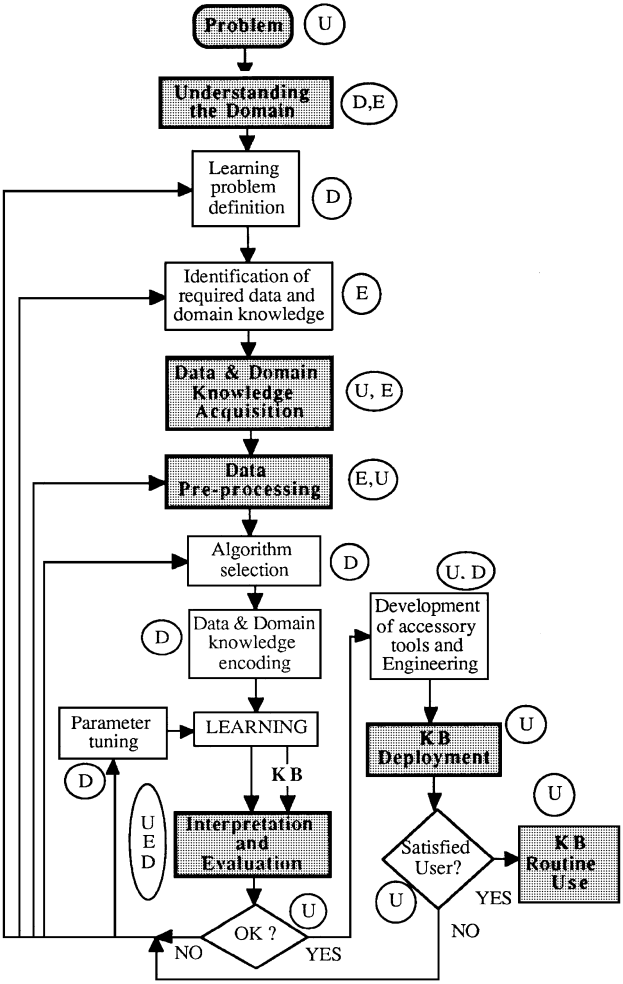
Problem
U
Understanding the Domain
D,E
Learning problem definition
D
Identification of required data and domain knowledge
E
Data & Domain Knowledge Acquisition
U, E
Data Pre-processing
E,U
Algorithm selection
D
U, D
Development of accessory tools and Engineering
D
Data & Domain knowledge encoding
Parameter tuning
LEARNING
KB
U
KB Deployment
D
U
E
D
Interpretation and Evaluation
U
Satisfied User?
KB Routine Use
U
YES
U
OK ?
NO
NO
YES

Figure 1. Scheme of the "collaborative" process of applying Machine Learning to a real-world problem. The participants in the process are the expert (E), the user (U) and the ML developer (D). The KB is the knowledge base outcome of learning. In each step, only the participant playing the essential role is explicitly indicated. Key issues are in bold and highlighted.

The graph of figure 1 does not capture the endless variety possible in real life; it only serves to focus the discussion on some major points. The first important observation is that at the origin of the whole process there is a user with a *problem*. The problem is one of the factors that Brodley & Smyth (1995) call “fixed” because they cannot be changed. Most often, the user presents the problem informally; thus the problem needs to be *conceptualized* and formalized. To this aim, the ML developer must become acquainted, with the help of the expert, with some background of the application domain (Asker & Maclin, 1997; Esposito, Malerba, & Semeraro, 1994). In other words, the situation referred to by Brodley (1997) as “give me a file of data and I will give you a bunch of rules” does not work in this kind of application.

Once the user’s problem has been clearly understood, it is not always obvious how to map it to a learning problem. Several researchers have reported on such a difficulty (see, for instance, Asker & Boström, 1995; Provost & Danyluk, 1995; Schwabacher, Hirsh, & Ellman, 1995). In many cases ML cannot solve the whole problem, and suitable subproblems must be identified, as in the case described by Schwabacher, Hirsh, & Ellman (1995).

The clarification of the original problem and the translation of it into a learning problem allows the information required for its solution to be identified. This information usually is gathered in the form of data and domain knowledge. According to our collaborative scheme, *Data acquisition* logically follows *Learning problem definition,* as also proposed by Brodley & Smyth (1995), Brodley (1997), Schwabacher, Hirsh, & Ellman (1995), and Provost & Danyluk (1995). In practice, it may well happen that data is already available and actually being the trigger of the learning process. This is the case of Data Mining, whose proposed life cycles (see, for example, those by Brachman et al., 1996; Fayyad, 1996; Fayyad, Haussler, & Stolorz, 1996; Fayyad, Piatetsky-Shapiro, & Smyth, 1996; Lindner & Klose, 1997; Adriaans, 1997) show “Data” at the very onset of the process.

Collecting data is a critical point, because data may not be available in a useful format, as Devaney & Ram (1997) report. Data, although available, may have to be labelled by the expert with additional effort, as in the application described by Provost and Danyluk (1995). People who have to do the practical work of collecting the data may not see any reason why they should do it. *Data pre-processing* includes cleaning the data, reducing the noise and deciding which features are relevant and which subset of the data to use. When the target problem lies at the knowledge frontier of a domain, the co-operation with a domain expert to select relevant features and/or subsets of the collected data may be the key to success, as Fayyad et al. (1995), Lindner & Klose (1997), Asker & Maclin (1997), and Esposito et al. (1990, 1997) note.

The choice of which specific learning algorithm to use may or may not be a critical step, depending on whether the intended-to-be-used algorithm is or is not developed by the ML researcher involved in the project, as it is often the case in this type of collaborative work (see, for example, Giordana et al., 1993; Bratko & Muggleton, 1995; Asker & Boström, 1995). In fact, one of the factors that drives researchers to invest time in an application is the desire to prove the usefulness of their own research. If this is not the case, exploration of alternative algorithms may become very time consuming, and systematic suggestions to help practitioners in this task would be highly welcome. An effort toward this end has been done by Sleeman et al. (1995) by developing the *Consultant* system. Langley (1997) and van Someren (1994), among others, report that in several comparative analyses a large variety of learning algorithms exhibited approximately equivalent behaviours. This finding contrasts with other cases in which substantial differences emerged among algorithms, as for instance, in the MLC++ project reported upon by Kohavi, Sommerfield, & Dougherty (1996). The issue of relating the learning algorithms to the type of data and to the nature of the problem to be solved still remains an open and fundamental problem. A possible way out may be the idea, still to be verified on large scale problems, of *combining* results from several algorithms rather than selecting a single learning system/approach (Freund & Shapire, 1996; Breiman, 1996; Asker & Maclin, 1997).

After running the learning algorithm and acquiring a body of knowledge, KB, the application process enters its most critical phase, namely, the *interpretation* and *evaluation* step. This phase involves all the participants in the project. In fact, the ML developer tries to understand whether the learning algorithm met the expectations. S/he will evaluate the results according to some precise metrics, for instance, error rate, CPU time, or number of rules or tree nodes. The expert will verify the compatibility of the results with available domain knowledge. But, at the bottom line, only the user is entitled to give the final judgement about the usefulness of the results. Brodley and Smyth (1995) note that the user is more attentive to *problem-dependent* criteria, often qualitative and subjective, which may be different, and often conflicting, with the previous ones. Schwabacher, Hirsh, & Ellman (1995), for example, used a problem-specific “design quality” criterion in the system they developed for engineering design optimization. Hence we disagree with Langley (1977), who says that evaluation has to be done by the ML developer “and possibly the expert.”

It is often mentioned that an important parameter in evaluating learned knowledge is *comprehensibility*. At the beginning of the development of the field, Michalski (1983) stated his *Comprehensibility Postulate*, and Michie (1982) argued in favour of introducing a “humanization loop” in an AI system architecture for achieving intelligibility. The debate is alive more than ever. The issue of comprehensibility has been the explicit subject of an IJCAI-95 Workshop (Nédellec, 1995), in which various facets of the term have been analyzed. The same stand is taken in the KA community, for instance by van Someren (1994). Even researchers working on *Explanation* in Expert Systems (Giboin, 1995, is an example) have tried to figure out if and how results in their field could be transferred to Machine Learning. Also in the new field of Knowledge Discovery in Databases (KDD) comprehensibility seems to be a defining issue. In fact, Fayyad (1996) states that

> “Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately *understandable* patterns in data.”

According to Craven & Shavlik (1995), comprehensibility is useful for knowledge validation (an expert wants to inspect rules to be confident in them (Giboin, 1995)), for discovery (knowledge has to be analyzed by human experts (Hunter & Klein, 1993)), for suggestions of better features and for knowledge refinement. Recently, Kohavi & Kuntz (1997) have advocated better comprehensibility as a motivation for proposing Option Decision Trees. Unfortunately, we do not have a principled mechanism for measuring comprehensibility. In his discussion on the topic, Kodratoff (1994a) points out the importance of the issue but notices that “The problem we are left with is that we do not understand comprehensibility!”

and advocates the status of an interdisciplinary “acknowledged research topic” for the study of comprehensibility. A different but related evaluation criterion has been brought up in the KDD community, i.e., the notion of knowledge *interestingness*. Silberschatz & Tuzhilin (1995) have even proposed a precise definition for it.

In order to be used, the acquired knowledge does not need to become the core of some computerized decision system. For instance, Evans & Fisher (1994) describe an application in which the learned knowledge had simply been written on a piece of paper for use by human operators. Typical, in this respect, is the knowledge discovery task: the expected product of learning is, in fact, “useful and interesting” information, to be used mainly by human decision makers.

Once preliminary testing is judged to be satisfactory, the learned knowledge base must be deployed for a period of in-field testing and, eventually, for routine use. Before deployment, the learned knowledge base may have to be embedded in a more complex system, which includes interfaces for human use, interfaces with other system modules, links with pre-existing databases and software, translation and/or compilation into more effective languages. This “engineering” step, represented in figure 1 as immediately preceding deployment, is actually an activity performed in parallel throughout the development of the application.

We stress explicitly that, in our opinion, a successful evaluation step is not yet sufficient for the application to be considered successful in the “real world.” In fact, this qualification can *only* come from lasting in-field use. This is the only proof of the effectiveness of the solution found with respect to the original problem. The situation is somewhat different when the product of learning is not a practical system but some piece of “hidden knowledge,” as in many KDD applications. In these cases, in-field use might mean that the discovered knowledge allowed some “good” course of action to be taken that would not have been possible without it.

The distinction between the test and the deployment phases is acknowledged also by Brodley & Smyth (1995), though they do not discuss the implications of this separation. Actually, very few systems generated with learning algorithms have been deployed and used in the field for a sizeable period of time. Most of the systems described by Langley & Simon (1995), Bratko & Muggleton (1995), Nakhaeizadeh (1995), or reported by Aha & Riddle (1995), Rudenström (1995) and Engels et al. (1997) did not go to the field, but remained in the phase between positive evaluation and deployment.

In fact, while progressing through the life cycle, assumptions and/or approximations have been most likely made about the domain, the problem, or the operational environment. For instance, the historical data used for training may have been cleaned, but raw data will be fed to the system in the field. “Real data is surprisingly dirty,” as noticed by Garner et al. (1995), the data is “extremely noisy and incomplete,” Devaney & Ram (1997) claim, or “real data is unreliable,” Provost & Danyluk (1995) warn. Often an application is first developed on simulated data as is the practice in dynamic system control. Even in the face of a very positive evaluation, the user may be reluctant to apply the learned controller on the real plant because even a small overlooked detail may have devastating effect, in terms of economical damages, or danger for human operators (see, for instance, Baroglio et al., 1996, and Giordana, 1994, for a discussion).

Finally, the operational environment is usually considered to be static, which is not realistic. For instance, Asker & Boström (1995) report that the system they built up did not succeed in the field because the training data was derived from too limited a time period that was not representative of long-term plant functioning. The same problem was reported by Nakhaeizadeh (1995) for two applications, one on fault diagnosis, and one on credit scoring, and also by Karba & Drole (1990).

In the collaborative-type life cycle most of the development time is not spent in letting the ML system run, but rather in the preparatory phase, especially in understanding the domain, defining the features and preparing the data. Engineering activities are fundamental to convince the user to accept and actually use the learned knowledge. Developing applications in the collaborative mode is time consuming; a project typically spans a range of several months to a couple of years.

3.2. “Stand-alone” process of application development

In figure 2, the life cycle of the “stand-alone” mode of applying ML to a problem is summarized. As the scheme shown in figure 2 is a subset of that in figure 1, much of the discussion from the previous subsection still holds and will not be repeated here. We will point out, instead, the shift of focus from the collaborative approach to the stand-alone one.

The stand-alone application type has the same input and output as the collaborative one, in the sense that the result of learning must be deployed, and the user must actively participate in the process. Actually, the user himself typically performs the whole process (possibly helped by a domain expert). This is the case when using off-the-shelf systems such as CART (Breiman et al., 1984), C4.5 (Quinlan, 1993) or one of the several Siftware tools for KDD available to potential users.[2]

The trigger of the learning process is still a problem that the user has to solve. This problem, in stand-alone applications, is usually sufficiently simple to be easily mapped onto a well-defined learning task, routinely solvable by commercially available systems. Maybe the user knows about Machine Learning and wants to explore some well-tested techniques. Because of this, the user’s commitment to the solution of the problem is weaker than in the collaborative case and s/he wants to invest in the project only a small amount of resources. For instance, Michie (1987) reports that American Express UK learned a decision tree for the problem of credit assignment. The application was developed in less than one week and brought a 20% accuracy increase as compared to the previously used method.

For the above reasons, the focus of the application development is not on problem conceptualization or data collection and pre-processing. Typically, the user exploits, in this kind of application, data already available in his/her company. For instance, s/he may want to mine an in-house database in search of possibly hidden useful information. The main focus here is on *algorithm selection* and a number of alternative learning techniques can be tried in turn.

In the *evaluation* step, important parameters are now *algorithm availability* and *simplicity of use*. Moreover, a friendly user interface and effective tools for data visualization are important, as noted by Schwabacher, Hirsh, & Ellman (1995). In fact, ML systems that are designed with this type of application process in mind typically have impressive interfaces,

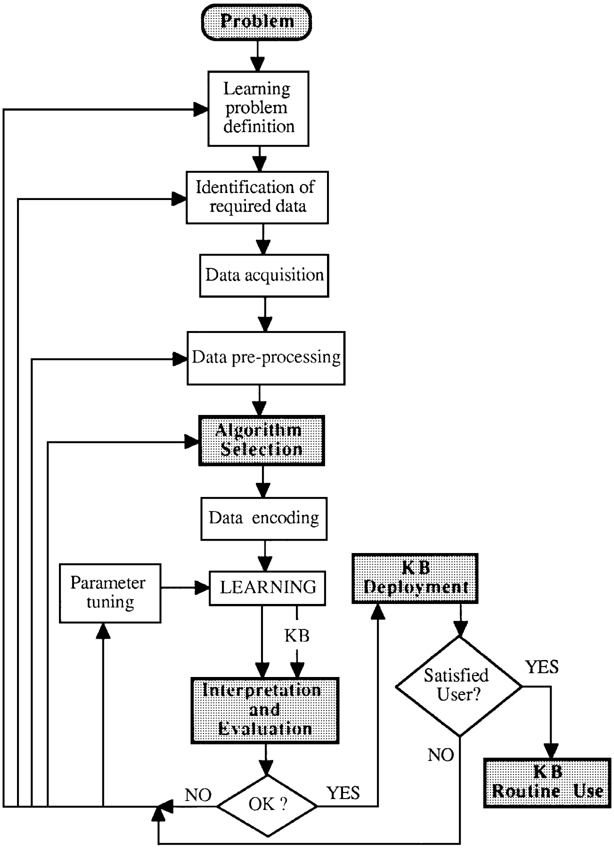

Figure 2. Scheme of the "stand-alone" process of applying Machine Learning to a real-world problem. Most often the user is the only participant.

as in the case of MineSet (Brunk, Kelly, & Kohavi, 1997), MultiSpec (Landgrebe & Biehl, 1994) or Clementine.[3] Note that the development of interfaces and the engineering phase disappeared from the cycle shown on figure 2 because this activity becomes an integral part of the ML developer's job, who aims at creating a ready-to-use, appealing product.

Between the two extreme scenarios described in figures 1 and 2, a wide spectrum of intermediate situations may occur (Garner et al., 1995). For instance, most of the schemes proposed for KDD (see, for example, Fayyad, Piatetsky-Shapiro, & Smyth, 1996) envision an interaction between the user and the ML developer, but not as close as the one required by the collaborative scheme.

4. "Ready-to-use" data sets

In order to test Machine Learning algorithms, it is common practice to use benchmark data sets such as those collected by Merz & Murphy (1997) in the Irvine repository. This “testing” procedure is not incorrect, *per se*, because evaluating algorithms on such data sets can actually provide important insights. However, what *is* wrong is to mistake these data sets for the “real world,” and to judge on their basis an algorithm’s practical adequacy. The simple fact that the names of the attributes have a meaning in the world does not automatically qualify a data set as a real application. Instead, Saitta, Giordana, & Neri (1995), and Kohavi & Kunz (1997) call these data sets “natural” in order to distinguish them from “artificial” ones.

In figure 3, we describe the steps commonly performed when learning from a ready-to-use data set. A specific (set of) ML algorithm(s) or method(s) is an input to the process and the goal is to investigate its properties. To this aim some data sets are chosen from the repository and are partitioned into training and test sets. The examples in a data set are already encoded, but occasionally some data pre-processing is needed in order to match the specific requirements of the algorithm(s). After learning, the obtained knowledge is evaluated on the test sets.

Fundamental differences appear between figures 1 and 2 on one side and figure 3 on the other. In figure 3 the *problem* is no longer an input to the process; furthermore, no deployment is planned and the results of the process are to be the subject of a scientific publication or technical report.

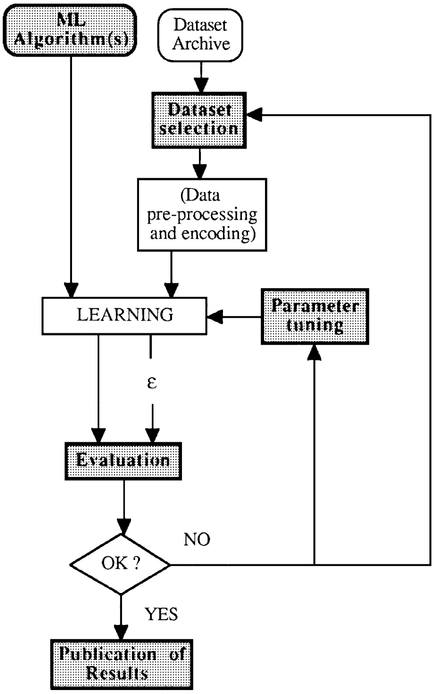

Figure 3. Life cycle of applying a ML algorithm to a ready-to-use data set. The symbol ε denotes a numerical evaluation of the results, most often an estimation of the error rate.

If the goal of the learning activity is to make some "objective" comparison among alternative algorithms or methods, the process is almost as linear as it has been described above, except possibly for a few loops through parameter tuning. If the goal of the work is to instead compare the author's new algorithm with competitors, it may happen that, in addition to a more accurate parameter tuning, other, better behaved data sets are selected. It is quite possible that this "algorithm tuning" is not intentional, perhaps not even conscious; it may simply be the result of a series of trials, some of which are more satisfying. This observation might be an explanation of a curious phenomenon: sometimes, each one of a set of algorithms turns out to be the "best" on the same data set when run by its creator(s).

Evaluation is most often based on *error rate*, and occasionally on some measure of syntactic simplicity. The learned knowledge base (KB) is usually not an outcome of the process. In fact, not only is the KB not intended for use, but it is almost never reported nor discussed, so it is impossible to assess its clarity, its adequacy to the problem, or its meaningfulness with respect to knowledge in the domain.

This being the case, one may wonder whether using ready-to-use data sets is of any utility. In fact, ready-to-use data sets do not reflect real life, because the learning task is usually vastly oversimplified, data is of limited size and complexity, difficult cases are often set aside, and knowledge of the domain is mainly ignored (Asker & Maclin, 1997). Satisfactory behaviour of a learning algorithm on some ready-to-use data set is not sufficient support for claiming its scalability to the real-world. Of course, this scalability is not excluded: simply, it should not be taken for granted.

The key issue is that testing an algorithm on standard benchmarks must not be a *goal* of the research, but only a *means*. Testing can be finalized to investigate the potential of an approach without investing too much effort and time, or to draw general conclusions. For instance, to know that algorithm A performed better than algorithm B on some data sets may not be of any utility for ML practitioners. On the contrary, an explanation of A's "better" behavior would shed light on the difficult question of relating classes of problems and well-suited types of algorithms (Brodley & Smyth, 1995). To this aim, standard benchmarks, precisely defined and correctly used, are fundamental.

Finally, we want to stress that the above criticism against ready-to-use data sets is not at all aimed at *theoretical* work: exactly the opposite is true, because they are criticized only if presented as real *applications*.

5. Excerpts from real-world applications

In this section, we begin with a brief survey of the systems we developed. Then we will describe four applications selected from the many we have addressed. These applications cover almost all the issues discussed in the previous sections, although they are related to very different domains. For the systems, we have chosen a historical perspective in order to follow the evolution of our research.

5.1. A history of system development

A time-ordered graph of the systems we developed over the years is shown in figure 4. We describe below the reasons behind each transition from an ancestor system to a descendant.

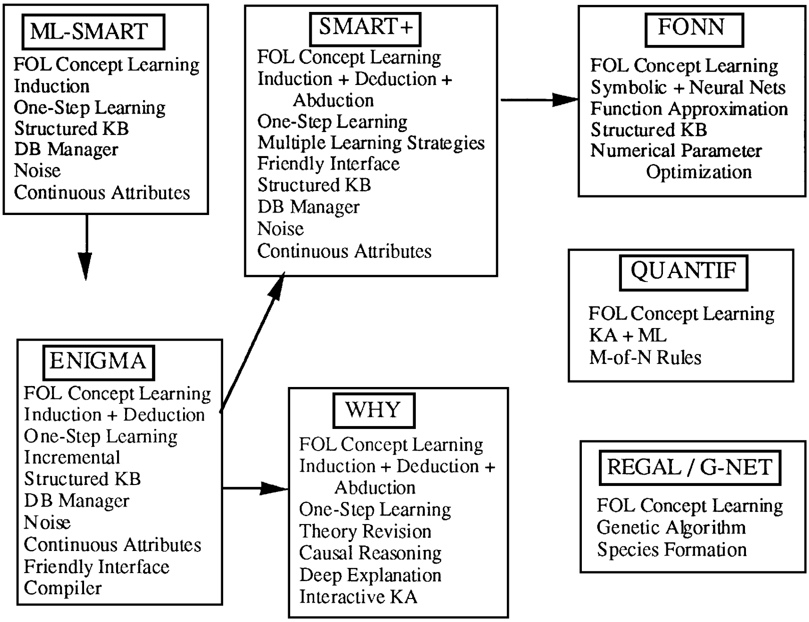

Figure 4. Overview of the systems we developed. Each arrow goes from an ancestor to a descendant system. The descendant inherits features from the ancestor and improves upon it along some dimension.

Our initial research in Machine Learning addressed the problem of learning classification rules expressed in First-Order Logic. Examples could be described by both categorical and numerical attributes and the presence of noise was explicitly taken into consideration (Bergadano, Giordana, & Saitta, 1988). The first system, ML-SMART, was an inductive learner. It adopted a top-down search strategy, acquired chained rules (not just a flat knowledge base) and performed discretization of continuous-valued attributes by means of fuzzy linguistic variables. The system was interfaced to an INGRES database in which both examples and extensions of hypotheses were stored.

The application of ML-SMART to the industrial troubleshooting problem described in Section 5.2 led to an interesting discovery: the human expert was not satisfied with an inductively derived knowledge base in spite of its strong empirical performance; he wanted to interpret the learned rules according to his own knowledge.

This need drove us to add to ML-SMART an EBL-like capability, which provided justification for the learned rules. This new version of ML-SMART was the system ENIGMA (Bergadano, Giordana, & Saitta, 1991). Prompted again by experts, we sought to provide a “qualitative” causal explanation for the learned rules. This research line resulted in the WHY system (Baroglio, Botta, & Saitta, 1994; Saitta, Botta, & Neri, 1993), which is able to learn diagnostic rules from examples and a causal model of the domain. This system was first applied to the same troubleshooting problem as ENIGMA, but later proved to be also well suited to a more human-oriented application, as described in Section 5.4.

On the other hand, for several applications it is important to be able to handle continuous numerical attributes while retaining a symbolic representation language. So ENIGMA evolved into SMART+ (Botta & Giordana, 1993), which has both locally optimized interval discretization and global optimization (by genetic search) of the numerical parameters occurring in rules. Furthermore, the industrial robotics application described by Baroglio et al. (1996) suggested to us forming a hybrid of the symbolic approach with the connectionist one. This hybridization led to the implementation of the system FONN (Botta, Giordana, & Piola, 1997).

As ML-SMART evolved into ENIGMA and then into WHY, it has become increasingly knowledge-intensive. Effective learning is achieved by exploiting large bodies of domain knowledge and more sophisticated reasoning tools. The sequence ML-SMART, ENIGMA, SMART+ and FONN, on the other hand, represents an increase in the ability to deal with noise, numerical aspects of the domain and interaction with databases.

Another development, driven more by scientific exploration than by concrete applications, was to follow an opposite trend, i.e., to limit the need for domain knowledge, and to increase the search power of the inference engine by using parallel genetic algorithms, as in the system REGAL, described by Giordana & Saitta (1994), and Giordana & Neri (1996), and in its recent successor G-NET, described by Anglano, Giordana, & Lo Bello (1997).

Finally, the system QUANTIF, which performs knowledge base restructuring by introducing M-of-N rules (Frediani & Saitta, 1986; Faure, Frediani, & Saitta, 1993), was developed in response to the need for this kind of representation in medical applications (Belforte et al., 1985).

5.2. Industrial troubleshooting application

The first large application we faced was to learn a diagnostic knowledge base for **mechanical troubleshooting**. In this application, which is a clear instance of the collaborative type, the systems ML-SMART, ENIGMA and WHY were employed. The user was the chemical company ENICHEM in Ravenna (Italy). The learned knowledge base has been used for five years by the company, both in the field and for training purposes. The parallel, manual development of an expert system devoted to the same task allowed the manual and the automatic generation of knowledge to be compared (Bergadano, Giordana, & Saitta, 1990).

At the ENICHEM chemical plant a mechanism was in place for performing predictive maintenance. All the electro-mechanical apparata undergo periodic predictive tests. The predictive maintenance process has two phases: a routine inspection and the *Mechanalysis*. During the inspection, pre-defined *routes* are followed daily by skilled personnel. Each route includes testing the operation parameters of a given set of machines. Should the technician detect, during the inspective control, something anomalous in the machine behaviour, the more accurate mechanalysis is performed. Mechanalysis is a very knowledge-intensive task, one at which very few diagnosticians excel. The knowledge is mostly derived from experience because the underlying quantitative models are too abstract and simplified to give concrete help in practice.

At ENICHEM, the mechanalyst in charge was retiring and the company had the problem of retaining his 20-year experience and the problem of training novices in the field without significantly increasing maintenance costs. Managers of the company knew of Expert

Systems and we were contacted because of our previous experience as knowledge engineers. The required diagnostic knowledge was different for different types of apparata, and the elicitation process would have to be repeated several times. An automatic acquisition process would have alleviated this problem, thus inspiring an interest in applying ML to this problem.

The 1500 electro-mechanical apparata, diagnosed by means of mechanalysis, range from small motor-pumps to very large turbo-alternators. All the apparata share the common feature of a rotating shaft to which various rotors are connected. When a fault occurs in the machine, anomalous vibrations appear. Mechanalysis performs a Fourier analysis of vibrations detected on the supports of the machine components. Each mechanalysis provides the basic features of a single example. The data is arranged into groups corresponding to the supports. Each group contains the measures of frequency and velocity of the harmonic components of the vibration for three spatial directions.

Automatically acquiring diagnostic knowledge from a set of previous mechanalyses is a difficult task for several reasons. First of all, the examples are complex, each one consisting of 20 to 60 items. An item is an entry in the mechanalysis table. This entry is the 4-tuple ⟨support, direction, frequency, velocity⟩ for each harmonic. Moreover, the examples are very noisy. In fact, all the measures are affected by large uncertainty margins, depending both on the intrinsic limits of the measurement apparata and on the human subjectivity in recording the observed values. Most of the mechanalyses performed during the 20 years of expert activity were available (about 600) but they had to be selected and cleaned (only about 300 remained).

For conceptualizing the learning problem, difficulties came from two main sources. On one hand, diagnostic classes were neither exclusive (multiple faults can be present at the same time with quite high probability) nor at the same level of abstraction (they were organized into a hierarchy, as shown in figure 5). Moreover, measured features turned out to be too low-level. The expert utilized intermediate concepts, defined as functions of basic features, and made strong use of the notion of context (presence of groups of features). For example, a particular frequency pattern (or value) may acquire great relevance in one context and may not be significant in another. The latter problem can be solved by defining a body of background knowledge that captures these intermediate-level features and using

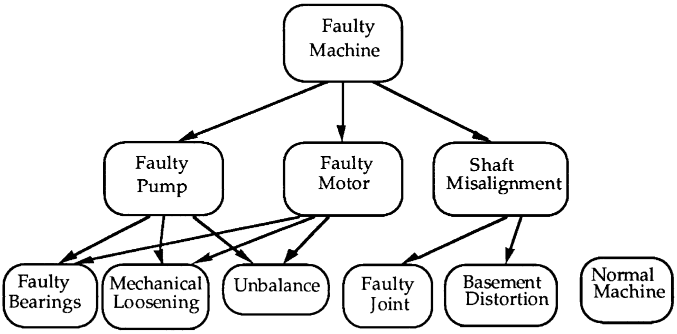

Figure 5. Taxonomic organization of the diagnostic classes for a motor-pump.

it to guide learning. The former problem requires adaptation of the learning algorithm, including a new definition of what an “error” is.

The first system we applied to the problem was ML-SMART, an inductive learning system. A difficulty arose in handling this application. Several of the learned rules, though good from the point of view of predictability, could not be understood by the domain expert. The expert was satisfied only if he could relate the automatically acquired knowledge to his own knowledge and understand the motivations underlying the rules. Statistical evidence was not always satisfactory as an explanation. So in the second phase the system ENIGMA, which exploits a body of background knowledge to guide induction, was applied to the same problem.

The rules acquired by ENIGMA are quite complex, but the expert evaluated them favourably because he could justify them in light of his mechanical knowledge. The expert was also pleased to observe that about one third of the learned rules coincided with those used by himself. He was also delighted to gain some new insights from the learned rules.

From the point of view of performance, the automatically acquired knowledge base appeared to be slightly superior to that elicited from the expert. This was confirmed by subsequent use in the field. The manually acquired knowledge base was totally replaced by the knowledge acquired automatically. It is interesting to note, however, that the system was accepted for use not because of its lower error rate, but because of its comprehensibility on the part of the expert.

The process of designing and implementing the manual expert system prototype took about 18 months, 12 of which were devoted to the acquisition, encoding and maintenance of the knowledge base. About three of the 12 months also served the task of defining the basic features, the conceptual model, and the domain theory to be used by ENIGMA. Performing several series of runs, under different operation conditions, took about two months.

The success of the application depended critically on two factors. The first was the knowledge of mechanical engineering possessed by one of the authors, which allowed her to interact with the expert. In fact, the expert had previously refused to co-operate with computer scientists on the basis of their inability to understand the domain concepts and terms. The second factor was the development of accessory modules such as a graphical interface, a compiler and a front-end to an information system where the diagnostic data was stored. The compiler allowed the logical format of the knowledge generated by ENIGMA to be translated automatically into the expert system shell GOLDWORKS, running on a PC, for use in the field.

Finally, the routine in-field use of the system led the users to ask for more. Was it possible to obtain not only the location and type of faults but also their plausible causes? This has been addressed with the development and subsequent application of the new system WHY. WHY’s causal explanation capability proved to be extremely useful for training purposes.

5.3. Helping the knowledge acquisition process

In this second ML application, human factors played a more fundamental role because the domain expert was an essential component of the learning cycle. The target of learning was a rule-based system for **speech spectrogram reading**, to be integrated with a fully automated expert system for speech segmentation and recognition. The user was a group at the E.N.S.T. in Paris (France) working on signal interpretation. The original task was one of knowledge acquisition (KA), i.e., elicitation from a domain expert of the rules she used for visual speech spectrogram segmentation. The interaction with the expert involved tape-recorded “thinking-aloud” sessions on specific cases with subsequent manual transcription and protocol analysis.

From the knowledge acquisition perspective, the task was difficult, because both “wired” cognitive routines and conscious reasoning were involved. This enabled the expert to verbalize only a part of her expertise. The elicitation task was boring for the expert because it was highly repetitive. Across different spectrograms the same phenomena occurred over and over again, but the expert wanted to mention them just once.

In order to cope with both these problems we used a mixed approach of KA and ML using the system QUANTIF, described by Frediani & Saitta (1986) and by Faure, Frediani, & Saitta (1993). QUANTIF’s knowledge representation language, i.e., “M-of-N” rules, was particularly well-suited to the task according to the expert. There were two reasons for choosing to apply a mixed KA and ML approach. The first was that we believed that ML could possibly capture the expert’s “unconscious” expertise by looking at her decisions. The second was that ML could be used to transform the repetitive knowledge elicitation task into one of knowledge revision, often much easier for an expert. The knowledge gaps left by the expert could be filled-in automatically and proposed to the expert for evaluation. The overall process was organized into a cycle, as shown in figure 6.

Before starting the cycle, the expert listed the features of the spectrogram she used for segmentation and for setting up the description language dictionary. Then she analyzed a number of spectrograms, corresponding to French sentences of 7–12 words, pointing out boundary locations between phonemes and the reasons for her decisions. The transcript from the protocol analysis was translated manually into a set of rules by the ML developer. In view of the subsequent automatic treatment, the manually translated rules were allowed to contain repetitions, redundancy, errors, or to be only partially specified. QUANTIF would then systematize and reorganize the rules into a coherent body to be submitted for revision to the expert. The process worked very well. The cycle was executed three times. At the end, the resulting knowledge base, satisfactorily tested on a set of new spectrograms, was embedded in the larger expert system.

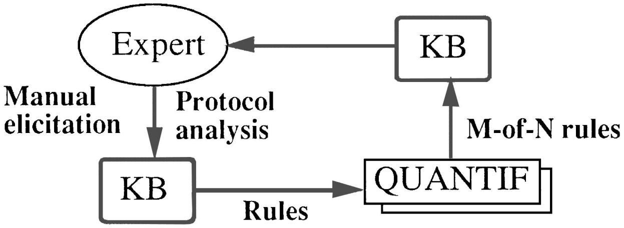

Figure 6. The cyclic interaction between manual elicitation and automatic knowledge restructuring.

This way of acquiring knowledge was advantageous because the expert was not required to be complete and consistent. An important result from this experience was the acquired awareness, on our part, of the difficulty of establishing the semantics of domain terms precisely, in such a way that an automatic system could make sense of them. In fact, during the first revision cycle it clearly emerged that many of the mistakes made by the first version of the rules were due to a slightly inconsistent semantic interpretation of the vocabulary terms by the expert. This fact did not hamper her performance because humans have a flexibility that artificial systems do not have. Without the help of QUANTIF it would have been much more difficult to locate these inconsistencies because the expert was not even aware of them.

The development of the application, starting from the phase of understanding the problem, until the moment in which the knowledge base was ready to be included in the expert system, took about four months. This application is another instance of the collaborative approach, even though not all the steps in figure 1 have been followed. In particular, data was already available and a basic understanding of the domain was acquired easily. On the other hand, it was not immediately clear how to use ML; hence the phase of defining the learning problem and the vocabulary (i.e., the features) turned out to be most critical. This application did not require development of particular accessory modules, but the learned knowledge was deployed and satisfactorily used as long as the expert system, in which it was embedded, has been used. The cycle of figure 5 instantiates the subcycle of figure 1 including *Data Pre-processing*, *Learning*, and *Evaluation*.

5.4. Modelling conceptual change in physics learning

A third, and rather unusual, application concerns using ML as a tool to model human learning. The particular application we describe is aimed at modelling the teaching/learning relationship when both the teacher and the learner are persons. Specifically, the context of the application, described by Neri, Saitta, & Tiberghien (1997), concerns teaching elementary physics concepts at a secondary school level.

In physics education studies, researchers such as Hestenes (1987) and diSessa (1996) have found that students show difficulties in learning at every school level, from elementary school to university. This phenomenon, specifically investigated by Tiberghien (1994), diSessa (1993), Vosniadou & Brewer (1994), and Slotta & Chi (1996), can be explained partially by the hypothesis that students have *misconceptions*, i.e., ideas formed from everyday experience, rooted mostly on perception, and contradicting accepted scientific theories. In order for students to overcome their misconceptions and to fully understand and accept “correct” theories, they must undergo a “conceptual change” (Carey, 1983; Smith, Snir, & Grosslight, 1992).

In our work, we have tried to model conceptual change with the learning system WHY, described by Saitta, Botta, & Neri (1993) and Baroglio, Botta, & Saitta (1994). We have exploited the conceptual framework introduced by Tiberghien (1994) and the results of her experimental work in a real teaching environment. The available data came from a group of eleven- and twelve-year-old students exposed to a physics course, including experimentation, questions, discussions and lectures. The course included basic concepts and qualitative relations in the domain of *heat transfer* in everyday life situations. The data consisted of the recorded answers for each student in personal interviews before and after the course, of questionnaires before and after the course, and of the transcripts of their videotaped answers to questions and discussions about experiments performed during each lesson. The user was a group of physics teachers, experimenting with new methods, whereas the expert was an educational scientist. The expert was also a kind of user because she wanted to exploit the model obtained to perform simulated teaching experiments.

The system WHY was judged well-suited to this task because its learning environment can account for all the relevant aspects of the theoretical framework proposed by Tiberghien. WHY learns and revises a knowledge base for classification problems. A complex inference engine combining induction, deduction, abduction and prediction is the core of the system, which is also able to provide explanations of the knowledge it learns in terms of a qualitative causal model of the domain.

WHY had two goals, when used to simulate the evolution of individual students: searching for evidence of conceptual change, and explaining “how” and “when” hypothesized conceptual change may have occurred. In order to use WHY to simulate a student’s evolution, each practical experiment or question played the role of an example consisting of two parts: a description of the experimental setting and a choice among possible outcomes or answers. Inferred from the material available from a student at the beginning of the course, a hypothetical body of knowledge has been encoded manually and inserted into WHY. Then, each time a new question or experiment is presented to the student, it is also presented to WHY, which updates its knowledge automatically, as the student does. The predictions and the explanations provided by WHY and the student are then compared. Different types of mismatch suggest the occurrence of different types of learning events in the student. An example of modelling is given by Neri, Saitta, & Tiberghien (1997).

This application is also an instance of the collaborative type, if we consider the expert as a user. Eventually, the targets of the application are both physics teachers and cognitive scientists. Teachers can use WHY to run simulations of individual students, to evaluate the expected effectiveness of teaching material and methods, and to (sub)optimally organize sequences of examples and notions to be taught. Cognitive and educational scientists can use WHY to test hypotheses quickly before setting up large in-field experimental work.

For this application the experimental data was already available. Most of the time was consumed in understanding the domain, identifying the learning problem, coding the causal knowledge, and interpreting the results. The learning problem was set up as one of theory revision. Not all the potential of WHY was exploited. In fact, some tasks that WHY could do automatically were performed manually, because in this phase the focus of the application was on discovering conceptual change, not on reproducing it automatically. As feedback from this application we received a confirmation that entirely new classes of challenging problems could be attacked by ML systems, should they become able to handle more elaborate types of background knowledge and to perform reasoning more sophisticated than simple induction.

5.5. *Splice sites recognition in a gene*

Recently, we became interested in Molecular Biology as a promising domain of application for Machine Learning. With the help of biologists, we succeeded in acquiring a basic understanding of the field and in figuring out what problems were of mutual interest. To start, we decided to apply our system REGAL, described by Giordana & Neri (1995), to the popular "Splice Junctions" data set contained in the Irvine repository. Even though the results we obtained were at that time better than all the other reported ones, as mentioned by Neri & Saitta (1996b), we discovered very soon that they were not useful for the corresponding "real" problem.

DNA is a double-helix shaped molecule composed of two strands. Each strand carries a linear sequence of four symbols (bases), namely the nucleotides A (Adenine), T (Thymine), C (Cytosine) and G (Guanine). A nucleotide strand contains sequences that "express" proteins ("coding sequences" or "genes"), as well as sequences that do not (Lewin, 1994). In eukaryotic organisms the coding part of the gene, copied onto the precursor RNA, is not a contiguous sequence but is spliced into subsequences, called *Exons*, interleaved with non-coding parts, called *Introns*, as shown in figure 7. The individuation of the boundary between an exon and an intron, called a *donor* site, and the one between an intron and an exon, called an *acceptor* site, can help in locating genes in unknown DNA strands.

Domain-specific knowledge states that splice sites almost always follow Chambon's "GT-AG" rule illustrated in figure 7. Moreover, specific "consensus" sequences occur around the boundaries (Mount, 1982; Breathnach & Chambon, 1981). Two acknowledged sequences are:

$$\binom{\mathrm{C}}{\mathrm{A}}\mathrm{AG}\mid\mathrm{GT}\binom{\mathrm{A}}{\mathrm{G}}\mathrm{AGT} \qquad \text{for a donor site} \tag{1}$$

$$\binom{\mathrm{T}}{\mathrm{C}}_n \times \binom{\mathrm{C}}{\mathrm{T}}\mathrm{AG}\mid\mathrm{G}\binom{\mathrm{G}}{\mathrm{T}} \quad (n \geq 11) \qquad \text{for an acceptor site} \tag{2}$$

In (1) and (2), the vertical bar denotes the boundary, which is preceded and followed by characteristic base sequences. The notation $(^{\mathrm{C}}_{\mathrm{T}})$ indicates that the nucleotides C or T are

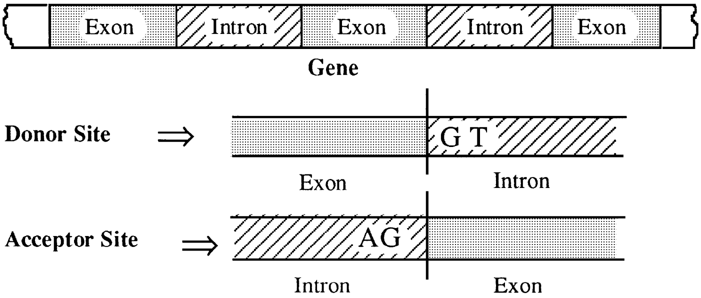

Figure 7. Splice sites inside a gene on a eukaryotic DNA. Most boundary sites follow the "GT-AG rule," which states that an intron begins with the dinucleotide GT and ends with the dinucleotide AG.

the most likely to occur in a specified location. X denotes any base. An intron starts at the bar in (1), and ends at the bar in (2).

5.5.1. The “splice junctions” data set. The “Splice Junctions” data set, as described by Noordewier, Towell, & Shavlik (1991), contains data from the GenBak 64.1, a repository of DNA sequences from various organisms, and consists of 3190 DNA sequences, each 60 nucleotides long. The problem of locating the sites of splicing in a gene has been mapped to a classification problem with three target classes: donor site (E/I class), acceptor site (I/E class), and no boundary (Neither class). The sequences in the data set are labelled according to these three classes. The proportions of representatives of the three classes are about 25%, 25% and 50%, respectively. The raw representation of each example consists of the sequence of identified nucleotides, starting at position -30 and ending at position $+30$ with respect to a possible boundary located between positions -1 and $+1$ (see figure 8).

Several ML algorithms have been run on this data set and a detailed comparison is reported by Neri & Saitta (1996b). The rules learned with our system REGAL showed, on the test set, a 3.42% error rate for the E/I class, 6.85% for the I/E class, and 4.29% for the Neither class. The system was run with a string of 60 bits, encoding 12 positions, from -6 to $+6$. To each position was associated one of the symbols {A, C, G, T, *}, corresponding to the base occurring at that position.[4] REGAL learned two sets of rules roughly conforming to consensus sequences (1) and (2).

5.5.2. Analysis of the results. The above reported results seem quite successful. However, they do not provide a solution to the original problem because the Irvine data set does not preserve the semantics of the real-world context.

First of all, the examples occurring in the data set, i.e., DNA segments 60 base long, containing or not containing a splice junction between positions -1 and $+1$, do not exist in the real world. Apart from the alignment problems (a fixed position for the splice site, which avoids the problem of the *reading frame*[5]), in the real setting only an unsegmented DNA string, several thousands of bases long, is available. A better test for the knowledge learned from the data set would be to try to identify splices in an unsegmented DNA string taken from one of the existing databases.

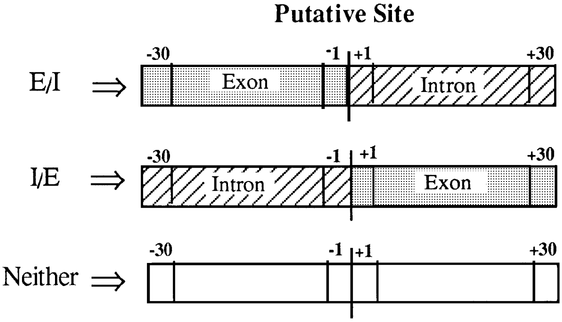

Figure 8. Examples of the E/I, I/E and Neither classes in the Irvine’s Splice Junctions data set. If either a donor or an acceptor site does exist, it is located between positions -1 and $+1$.

Moreover, the class distribution of the original classification problem is highly unbalanced; no more than 6% of the DNA is estimated to code for human genes (Lewis, 1994, Chapter 23). In a given string, there are very few positive examples and a very large number of negative ones, which can be a problem for ML algorithms, as noticed by Kubat & Matwin (1997). The Irvine data set has a ratio of negative (Neither) to positive examples (I/E $+$ E/I) of $1 : 1$, which is far from reality.

The "real" problem requires a much more sophisticated approach. For instance, the system GRAIL (available over the Internet and well known in the Molecular Biology community) uses a $20 : 1$ ratio between negative and positive instances for training. It takes a multi-agent approach, which exploits several classification modules (agents) interacting hierarchically (Uberbacher & Mural, 1991; Xu et al., 1996). GRAIL operates by filtering, in cascade, potential coding regions and ranking DNA substrings according to a model of the gene. Two neural networks suggest, separately, the presence of an acceptor or a donor site on the basis of seven features measured on the strings. The outputs of the networks are given as input, together with other parameters, to another neural net, whose output is a final score of the substring as a possible exon. Information on Expressed Sequence Tags (Xu, Mural, & Uberbacher, 1996) has recently been added. One must conclude that the result of learning on the Irvine data set can contribute, at best, to a very small subproblem of that which must be solved.

The knowledge base learned by REGAL from the Irvine data set was used to predict splice sites on 25 human DNA sequences downloaded from GenBank. For the sake of illustration, we report here the results obtained on the β-globin gene cluster. The GenBank's sequence in which the β-globin cluster occurs is 73,308 base long, and contains five genes (ε, Gγ, Aγ, δ and β) and one pseudo-gene (ψ-β). Each gene of this cluster has exactly three exons and two introns; the global number of splice sites is then 12 donors and 12 acceptor sites. Ignoring the reading frame, we can let a 12-base-long window slide along the DNA string one base at a time and we can evaluate how many window positions satisfy the rules for a donor or an acceptor site between position -1 and $+1$. The result is that 45 positions satisfy the rules for donor sites and 165 positions satisfy the rules for acceptor sites. All the 72,787 other positions are classified as Neither. Six of the donor sites and four of the acceptor sites are correctly individuated, giving 0.06% error rate for the E/I class and 0.2% error rate for the I/E class. If we consider as examples only those windows in which the "GT-AG" rule is verified (3334 for the donor sites and 4714 for the acceptor sites), the corresponding error rates become 1.3% for the E/I class and 3.6% for the I/E class. These error rates are low but are not as good as they look. This is because they are mostly due to the overwhelming presence of examples of the Neither class. In fact, only 10 of the 24 splice sites have been correctly identified (58% errors of omission), whereas 200 false sites have been detected. These figures were judged by the experts of the domain to be far from acceptable.

On the other hand, the same domain experts were impressed by the ability of REGAL to re-discover the known consensus sequences (and also some other features, such as the "TATA box," in the "promoter" region). They decided it worthwhile to start a closer cooperation in a more realistic setting. In fact, we have started a more sophisticated learning approach, described by Calapaj, Lo Bello, & Saitta (1997), using unsegmented sequences taken from human genome repositories.

6. Synoptic view of the emerged issues

In this section we summarize the issues that emerged in the previous ones, providing a tentative categorization of the different modes of performing ML applications and comparing their relative merits with respect to feedback to ML research.

ML applications show a continuously varying degree of situatedness in the world according to a number of aspects. We will take into consideration two of them, which we feel are key factors in defining the “real-worldliness” and the impact on future research of an application. The first is the degree of interaction between the Machine Learning developer on one side and the expert/user on the other. The second is the user’s commitment to solve a specific problem, and its impact on the kind of evaluation the learned knowledge receives. These two dimensions are represented on the axes of figure 9. As we will elaborate in the following, the “real-worldliness” of an application increases from bottom to top, whereas the impact to future research increases from left to right. As a consequence, we (arguably) claim that the most rewarding region to work in is the top-right one.

By combining the two dimensions, nine regions can be identified in figure 9. Three of them need not be considered: the left-top region is an oxymoron, so it has no instances. The center-bottom and right-bottom ones are uninteresting, because a user most likely will not explore ML on data sets *only*. This activity may be part of a project, in which case it is included in some other region of the graph. Moreover, users typically use in-house data.

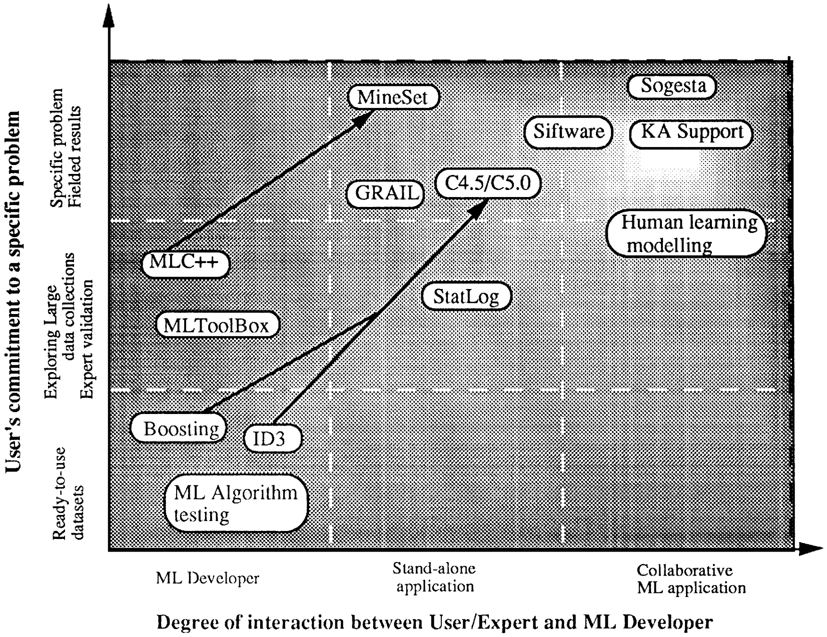

Figure 9. Classification of ML application types according to two dimensions: the level of user’s commitment to the solution of a problem, and the degree of interaction between ML developer and user.

The left-bottom region is “algorithmic-testing centered” and contains most of the experimental work in ML. Here an algorithm’s properties are studied by means of benchmark data. An instance of this region has been discussed in Section 5.5 (Splice Junction recognition from Irvine’s data set). In this region, the evaluation of the results is provided only by the ML researcher. Work in this region might be a preliminary step in the development of methodologies scaleable later to real-world problems. Examples of this evolution are C4.5 (Quinlan, 1993) and MineSet (Brunk, Kelly, & Kohavi, 1997). The opposite may also happen: data used in a real application are made available to other researchers through some data set collection. However, the data loses most of its real-worldliness in the transfer.

The left-center region contains those cases in which ML researchers have asked an external source to provide some set of data more “realistic” for its size and complexity. The data provider will be asked to give his/her opinion on the results, i.e., an a posterior evaluation of the outcome of the learning process. The real-worldliness degree of this region is higher than that of the bottom-left region, but not enough to label it as containing real-world applications. All the cases in which ML developers test their algorithms on “natural” data fall in this region. A case that well exemplifies this region is the MLToolBox project (see, for example, Sleeman et al., 1995).

The center-center region contains those ML applications where a user carefully evaluates alternative ML tools, usually with the aim of exploring their potential. Evaluating the results may require a big effort in terms of both costs and human resources. Research work such as that reported in the StatLog project (Michie, Spiegelhalter, & Taylor, 1994) mainly belongs to this region.[6] The real worldliness level of these applications is higher than the ones of all the previously discussed regions. However, this region still does not contain “real-world” ML applications because no one is using the learned knowledge.

The center-top region contains successful ML applications performed by the user him/herself. These applications are “real-world” ones according to our definition. This region contains applications performed using systems like CART (Breiman et al., 1984), C4.5 (Quinlan, 1993) or MineSet (Brunk, Kelly, & Kohavi, 1997) to solve specific users’ problems.

The right-center region contains those ML applications that are characterized by an active involvement of the user, the expert and the ML developer to solve some problem. In such situations, a careful user evaluation of the learned knowledge also must take place. However, the in-field deployment phase may not be reached, possibly because the learned knowledge still does not adequately solve the given problem or because some additional work is required before deployment. An instance of this type of application has been discussed in Section 5.4, i.e., the human learning modelling.

The right-top region clearly contains real-world ML applications that owe their success to the active involvement, from the beginning, of a domain user/expert to solve a problem. The industrial troubleshooting application, described in Section 5.2, and the speech recognition application, described in Section 5.3, belong to this region, as well as all the fielded applications cited by Langley & Simon (1995) and by Ruderström (1995).

By summarizing, the graph can be discussed according to its horizontal or vertical partition. A ML application reaches a specific horizontal level depending upon the point at which its developing cycle has stopped or to which it can be extended. For instance, an algorithmic-centered ML application (left-bottom) can be extended to have a posterior user/expert evaluation (left-center), but it can also reach the level of user-evaluated application (center-top or right-top). Hence, the left-bottom region should be considered only as a transient phase in the research.

The stand-alone and the collaborative approaches naturally reside in the center-top and right-top regions, if carried out through the in-field deployment. If this phase is not reached, they may stop at a lower level.

The reason for introducing a distinction between the two top-most horizontal strips is that a positive evaluation on the part of the user does not automatically imply knowledge *acceptability*, as Forsyth & Rada (1986) note. There is still a gap between the expert’s approval of the learning results and the actual use of them. A user may be impressed by the fact that an artificial system was able to rediscover what s/he had learned before, or even that it provided something new, but this may not be sufficient to determine a change in his/her everyday practice. In fact, to change working habits and to accept technological innovations may be a very demanding task, both in terms of concrete costs and cognitive efforts.

Considering the vertical partition of the graph, we feel that activity limited to the leftmost column (and especially to the left-bottom region) has a low potential to stimulate advances in a domains such as Machine Learning. In fact, ML algorithms are not “correct” or “incorrect,” but only “useful” or “not useful.” Hence, their ultimate evaluation must come from outside the ML community. According to our experience it is possible that theoretical work may lead to novel solutions to real-world problems, but we believe that it is more likely that real-world problems themselves will lead to those solutions faster if ML researchers work on them directly.

The role of the central column is more controversial. We have noticed that users that experiment with ML freeware tend not to give feedback to the product developer. For the ML researcher, then, a stand-alone application may not be rewarding in terms of new research stimuli derived from the use of the product; hence, Foley (1996) notes, valuable experience may get lost. On the other hand, if a user buys a ML tool, it may happen that s/he reports to the developer the results of his/her experience, frequently asking for improvements. However, the nature of these new requests is more related to improvements in the facility of use, such as better interfaces, more menu choices, more effective ways of memorizing data, and so on. This type of user seldom proposes new challenges to ML methodologies. However, it is possible that our experience is specific for Italy (or Europe).

A counterexample of this claim is probably the case of data mining. Potential users of available ML and statistical tools failed in using them on large amounts of data, suggesting thus that scalability with respect to data size was a challenge to learning methods. Also Nakhaeizadeh (1995) acknowledges feedback for Daimler-Benz’s applied ML research from the StatLog project.

The rightmost column, in which there is an active participation of the user/expert from the beginning, is the most fruitful one because both parties (user and ML developer) get the highest benefits. The user has a fielded application that solves his/her problem, and the ML researcher is likely to receive extensive feedback to guide his/her future research. It has to be said that fruitful hints for research are not just request to fix some bugs, but they should allow researchers to generalize the suggested issues in order to address a larger class of problems and/or situations, possibly leading to new approaches and methodologies.

7. Conclusions

In this paper we have discussed some issues related to real-world applications of ML, based both on our experience and on that reported by other researchers. By applying ML to complex real-world problems, we have discovered that the most interesting research challenges are very often *application-driven*. Namely, they spontaneously emerge when some still unsolved problem must be addressed. This scientific feedback obtained may involve the need for slightly adjusting the set of facilities provided by a learning system, the development of some new algorithm (e.g., Buntine & Patel, 1995), or even the creation of a new research field, as in the recent case of KDD.

We feel that there are many advantages in shifting toward a problem-driven mode of applying ML. Piatetsky-Shapiro (1997) reported a similar trend in the KDD community. First generation KDD systems were "research-driven," were oriented toward the expert, and required a deeper understanding of their functioning. Second generation systems were "vendor-driven," composed of a variety of tools (in a single system) and were oriented to more autonomous use. Third generation systems show a shift toward "application-specificity" by providing a shell that the user can customize according to specific needs.

Langley & Simon (1995) assert that Machine Learning "has replaced knowledge engineering with two simpler tasks: characterizing the problem and designing a good representation." However, people working in KA have noticed that the discipline still involves programming and representation languages for which no ML systems are available. Furthermore, a purely inductive approach is often not feasible for KA, van Someren (1994) claims. Moreover, experts and knowledge engineers have often found that the ML approach to KA was "simple-minded." For instance, Clancey (1983) notices that "a simple rule based system does not support the generation of adequate explanations," and that the performance of automatically induced knowledge bases is often comparable with average experts, not the outstanding ones.

Interdisciplinary work is always a social act. Sometimes the success or failure of a project does not depend as critically on technical issues as it does on psychological or social ones (Brodley & Smyth, 1995; Garner et al., 1995). Isaacs & Tang (1996), in their analysis of technology transfer, clearly state the relevance of these aspects and point out a few basic ingredients for the success of projects: research people highly motivated to see their ideas concretized, readily demonstrable improvements over existing products, and customers with a deep need for a new technology. As already mentioned, in complex applications a lot of time is spent in developing accessory modules, such as user-friendly interfaces (Langley & Simon, 1995), or links to existing software or hardware (van Someren, 1994). For researchers, it may not be rewarding to devote time to produce software as a product. On the other hand, even the most effective and clever method may not have a chance of getting known or used if nobody can use it except its developer.

In our experience of working on real-world projects, we met people whose attitude toward ML varied broadly. Some were just interested in labelling their products with "Contains Machine Learning" (i.e., "The best cutting-edge technology you can buy"), with the lowest possible cost and effort. Others were experts in some areas (medicine, for instance), who had heard of ML and wanted to try the methodology, half out of curiosity and half in the hope of obtaining non-trivial new information from their data. A third type of potential ML customers were people strongly motivated to make a substantial change in their currently used technologies, and determined to invest time and resources for it. In accord with Isaacs and Tang (1996), we also believe that researchers in academia should favor the collaborative way of interaction.

Acknowledgments

We are deeply indebted to all the current and former members of the Machine Learning group at the Dipartimento di Informatica of the University of Torino, who devoted time and efforts to the development of ML systems and to their application. We also want to thank Foster, Ronny and four anonymous reviewers, who worked on this paper almost as much as we did. Finally, we thank the European Science Foundation for the support given to the project “Learning in Humans and Machines,” in whose context the work described in Section 5.4 has been performed.

Notes

1. The present paper is an extended version of the one by Saitta, Giordana, & Neri (1995).
2. A list of Siftware tools can be found at http://www.kdnuggets.com/siftware.html.
3. A reference to Clementine can be found at http://www.isl.co.uk/clem.html.
4. There is a 1% occurrences of the symbols X (denoting any base) and D (denoting A or G or T), when nucleotides are recognized ambiguously. The symbol “*” is used when either an X or a D occurs in the sequence.
5. As each amino acid is codified by a triplet of bases, there are three ways of “reading” a DNA strands, each one shifted 1 base to the right. Only one of these “reading frames” corresponds to the correct coding of the gene.
6. A part of the StatLog project belongs to the left-center region, because ML developers from universities also contributed. We consider, here, the industrial component of the project.

References

Adriaans, P. (1997). Industrial requirements for ML application methodology. *Proceedings of ICML-97 Workshop on Machine Learning Applications in the Real World; Methodological Aspects and Implications* (pp. 6–10).

Aha, D., & Riddle, P. J. (Eds.) (1995). *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Anglano, C., Giordana, A., Lo Bello, G., & Saitta, L. (1997). C-GNET: A coevolutionary approach to concept-induction. *Proceedings of the Seventh International Conference on Genetic Algorithms* (pp. 434–441). San Francisco, CA: Morgan Kaufmann.

Asker, A., & Boström, H. (1995). Building the DeNOx system: Experience from the real-world application of machine learning. *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Asker, L., & Maclin, R. (1997). Feature engineering and classifier selection: A case study in Venusian volcano detection. *Proceedings 14th International Conference of Machine Learning* (pp. 3–11). San Francisco, CA: Morgan Kaufmann.

Baroglio, C., Botta, M., & Saitta, L. (1994). WHY: A system that learns from a causal model and a set of examples. In R. Michalski, & G. Tecuci (Eds.), *Machine Learning: A Multi-Strategy Approach* (Vol. 4). San Francisco, CA: Morgan Kaufmann, pp. 319–347.

Baroglio, C., Giordana, A., Kaiser, M., Nuttin, M., & Piola, R. (1996). Learning controllers for industrial robots. *Machine Learning, 23,* 221–250.

Belforte, G., Bona, B., Cravetto, C., Frediani, S., Milanese, M., Molino, M., Saitta, L., & Tempo, R. (1985). Selection and assessment of laboratory tests for the evaluation of liver functional impairment. *Methods of Information in Medicine*, *24*, 39–45.

Bergadano, F., & Giordana, A. (1988). A knowledge intensive approach to concept induction. *Proceedings of 5th International Conference in Machine Learning* (pp. 350–317). San Francisco, CA: Morgan Kaufmann.

Bergadano, F., Giordana, A., & Saitta, L. (1988). Learning concepts in noisy environment. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-10,* 555–578.

Bergadano, F., Giordana, A., & Saitta, L. (1990). Automated versus manual knowledge acquisition: A comparison in a real domain. *Proceedings of First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop* (pp. 301–314).

Bergadano, F., Giordana, A., & Saitta, L. (1991). *Machine Learning: A General Framework and its Application*. Chichester, UK: Ellis Horwood Ltd.

Blanzieri, E., & Katenkamp, P. (1996). Learning radial basis function networks on-line. *Proceedings of 13th International Conference in Machine Learning* (pp. 37–45). San Francisco, CA: Morgan Kaufmann.

Botta, M., Saitta, L., Brancadori, F., De Marchi, D., & Radicchi, S. (1991). Automatic construction of second generation diagnostic expert systems. *International Journal of Expert Systems*, *4*, 389–400.

Botta, M., & Giordana, A. (1993). Smart+: A multi-strategy learning tool. *Proceedings 13th International Joint Conference on Artificial Intelligence* (pp. 937–943). San Francisco, CA: Morgan Kaufmann.

Botta, M., Giordana, A., & Piola, R. (1997). FONN: Combining first order logic with connectionist learning. *Proceedings of 14th International Conference on Machine Learning* (pp. 48–56). San Francisco, CA: Morgan Kaufmann.

Brachman, R. J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G., & Simoudis, E. (1996). Mining business databases. *Communications of ACM*, *39*(11), 42–48.

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

Breathnach, R., & Chambon, P. (1981). Organization and expression of eucaryotic split genes coding for proteins. *Annual Review of Biochemistry, 50,* 349–383.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth.

Brodley, C. (1997). The need for diagnostics for classification algorithms. In R. Engels et al. (Eds.), *Proceedings of ICML-97 Workshop on Machine Learning Applications in the Real World; Methodological Aspects and Implications* (pp. 12–14).

Brodley, C., & Smyth, P. (1995). The process of applying machine learning algorithms. *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023, pp. 7–13). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Brunk, C., Kelly, J., & Kohavi, R. (1997). MineSet: An integrated system for data mining. *Proceedings of 3rd International Conference on Knowledge Discovery and Databases* (pp. 135–138). Menlo Park, CA: AAAI Press.

Buntine, W., & Patel, T. (1995). Intelligent instruments: Discovering how to turn spectral data into information. *Proceedings of the First International Conference on Knowledge Discovery and Databases* (pp. 33–38). Menlo Park, CA: AAAI Press.

Calapaj, L., Lo Bello, G., & Saitta, L. (1997). *Genetic algorithm-based constructive induction for the analysis of promoter regions in DNA* (Technical Report TR 1202-97). Torino, Italy: Universita di Torino, Dipartimento di Informatica.

Carey, S. (1983). *Conceptual Change in Childhood*. Cambridge, MA: MIT Press.

Craven, M. W., & Shavlik, J. S. (1995). Extracting comprehensible concept representations from trained neural networks. In C. Nédellec (Ed.), *Proceedings of IJCAI-95 Workshop on Machine Learning and Comprehensibility* (pp. 61–75).

Devaney, M., & Ram, A. (1997). Situation development in a complex real-world domain. In Engels et al. (Eds.), *Proceedings of ICML-97 Workshop on Machine Learning Applications in the Real World; Methodological Aspects and Implications* (pp. 43–47).

diSessa, A. (1993). Toward an epistemology of physics. *Cognition and Instruction, 10*, 105–225.

diSessa, A. (1996). What do "just plain folk" know about physics. In D. Olson, & N. Terrance (Eds.), *Handbook of Education and Human Development*, Blackwell Publishers, pp. 709–730.

Engels, R., Evans, B., Herrmann, J., & Verdenius, F. (Eds.) (1997). *Proceedings of ICML-97 Workshop on Machine Learning Applications in the Real World; Methodological Aspects and Implications*.

Esposito, F., Malerba, D., Semeraro, G., Annese, E., & Scafuro, G. (1990). Empirical learning methods for digitized document recognition: An integrated approach to inductive generalization. *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications* (pp. 37–45). Los Alamitos, CA: IEEE Computer Society Press.

Esposito, F., Malerba, D., & Semeraro, G. (1994). Multistrategy learning for document recognition. *Applied Artificial Intelligence*, *8*, 33–84.

Esposito, F., Malerba, D., Semeraro, G., Fanizzi, N., & Ferilli, S. (1997). Adding intelligence to digital libraries: IDL. *Proceedings of the IJCAI-97 Workshop on AI in Digital Libraries* (pp. 23–31). San Francisco, CA: Morgan Kaufmann.

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

Faure, C., Frediani, S., & Saitta, L. (1993). A semiautomated methodology for knowledge acquisition. *IEEE Transactions Systems, Man, and Cybernetics, SMC-23*, 346–356.

Fayyad, U. (1996). Data mining and knowledge discovery: Making sense out of data. *IEEE Expert*, *11*(5), 20–25.

Fayyad, U., Haussler, D., & Stolorz, P. (1996). KDD for science data analysis: Issues and examples. *Proceedings of the 2nd Knowledge Discovery and Data Mining Conference* (pp. 50–56). Menlo Park, CA: AAAI Press.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of ACM*, *39*(11), 27–34.

Foley, J. (1996). Technology transfer from university to industry. *Communications of the ACM*, *39*(9), 30–31.

Forsyth, R., & Rada, R. (1986). *Machine Learning: Applications in Expert Systems and Information Retrieval*. Chichester, UK: Ellis Horwood.

Frediani, S., & Saitta, L. (1986). Knowledge base organization in expert systems. *Proceedings of Conference on Uncertainty in Knowledge-Based Systems* (pp. 217–224). LNCS 286, Berlin, Germany: Springer-Verlag.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the 13rd International Conference of Machine Learning* (pp. 149–156). San Francisco, CA: Morgan Kaufmann.

Garner, S. R., Cunningham, S. J., Holmes, G., Nevill-Manning, C. G., & Witten, I. H. (1995). Applying a machine learning workbench: Experience with agricultural databases. *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023, pp. 14–21). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Gemello, R., & Saitta, L. (1987). *Automated acquisition of phonemic-phonetic transcription rules* (CSELT's Technical Reports, Vol. 16, No. 7). Torino, Italy: CSELT.

Gemello, R., Mana, F., & Saitta, L. (1991). Rigel: An inductive learning system. *Machine Learning*, *6*, 7–36.

Giboin, A. (1995). ML comprehensibility and KBS explanation: Stating the problem collaboratively. *Proceedings of IJCAI-95 Workshop on Machine Learning and Comprehensibility* (pp. 1–12). San Francisco, CA: Morgan Kaufmann.

Giordana, A. (1994). Applications of machine learning to robotics. *Proceedings of MLNet Workshop on Industrial Applications of Machine Learning* (pp. 59–68). Dourdan, France.

Giordana, A., & Sale, C. (1992). Genetic algorithms for learning relations. *Proceedings 9th International Conference on Machine Learning* (pp. 169–178). San Francisco, CA: Morgan Kaufmann.

Giordana, A., Saitta, L., Bergadano, F., Brancadori, F., & De Marchi, D. (1993). ENIGMA: A system that learns diagnostic knowledge. *IEEE Transactions on Knowledge and Data Engineering, KDE-5*, 15–28.

Giordana, A., & Saitta, L. (1994). Learning disjunctive concepts by means of genetic algorithms. *Proceedings of International Conference on Machine Learning* (pp. 96–104). San Francisco, CA: Morgan Kaufmann.

Giordana, A., & Neri, F. (1995). Search intensive concept induction. *Evolutionary Computation*, *3*, 375–416.

Giordana, A., Neri, F., Saitta, L., & Botta, M. (1997). Integrating multiple learning strategies in first order logics. *Machine Learning*, *27*, 209–240.

Hestenes, D. (1987). Toward a modelling theory of physics instruction. *American Journal of Physics*, *55*, 440–454.

Hunter, L., & Klein, T. (1993). Finding relevant biomolecular features. *Proceedings of 1st International Conference on Intelligent Systems for Molecular Biology* (pp. 190–197). Menlo Park, CA: AAAI Press.

Isaacs, E. A., & Tang, J. C. (1996). Technology transfer: So much research, so few good products. *Communications of ACM, 39*(9), 23–25.

Karba, N., & Drole, R. (1990). Expert system for the cold rolling mill of the steel works Jesenice. *Proceedings of the 13th Symposium on Information Technologies,* Sarajevo.

Kodratoff, Y. (1994a). *AI Communications, 7*(2), 83–85.

Kodratoff, Y. (Ed.) (1994b). *Proceedings of MLNet Workshop on Industrial Application of Machine Learning*. Dourdan, France.

Kohavi, R., & Kunz, C. (1997). Option decision trees with majority votes. *Proceedings of 14th International Conference in Machine Learning* (pp. 161–169). San Francisco, CA: Morgan Kaufmann.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1996). Data mining using MLC++: A machine learning library in C++. *Proceedings of Tools with Artificial Intelligence* (pp. 234–245). Los Alamitos, CA: IEEE Computer Society Press.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. *Proceedings of 14th International Conference on Machine Learning* (pp. 179–186). San Francisco, CA: Morgan Kaufmann.

Landgrebe, D., & Biehl, L. (1994). *An Introduction to MultiSpec.* Lafayette, IL: Purdue Research Foundation.

Langley, P. (1997). Challenges for the application of machine learning. In Engels et al. (Eds.), *Proceedings of ICML-97 Workshop on Machine Learning Applications in the Real World; Methodological Aspects and Implications* (pp. 15–18).

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

Létourneau, S., Famili, A., & Matwin, S. (1997). Discovering useful knowledge from aircraft operation/maintenance data. *Proceedings of the ICML-97 Workshop on Machine Learning Applications in the Real World: Methodological Aspects and Implications* (pp. 34–41).

Lewin, B. (1994). *Gene V.* New York, NY: Oxford University Press.

Lindner, G., & Klose, A. (1997). ML and statistics for trend prognosis of complaints in the automobile industry. In Engels et al. (Eds.), *Proceedings of the ICML-97 Workshop on Machine Learning Applications in the Real World: Methodological Aspects and Implications* (pp. 20–26).

Merz, C. J., & Murphy, P. M. (1997). *UCI Repository of Machine Learning Databases*. Dept. of Information and Computer Science, University of California at Irvine, Irvine, CA. http://ics.uci.edu/pub/machine-learning-databases.

Michalski, R. (1983). A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 1). San Francisco, CA: Morgan Kaufmann, pp. 83–134.

Michie, D. (1982). *Machine Intelligence and Related Topics*. New York, NY: Gordon and Breach Science Publishers.

Michie, D. (1987). Problems of computer-aided concept formation. In J. R. Quinlan (Ed.), *Applications of Expert Systems* (Vol. 2). Wokingham, UK: Addison-Wesley.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. UK: Prentice Hall.

Milanese, M., Bona, B., Frediani, S., Saitta, L., Cravetto, C., & Molino, G. (1982). Optimization of diagnostic procedures in hepatology. *Proceedings of the First Annual Conference American Association for Medical Systems and Informatics* (pp. 19–22).

Mitchell, T. (1997). Does machine learning really works? *AI Magazine, 18*(3), 11–20.

Mount, S. M. (1982). A catalogue of splice junction sequences. *Nucleic Acid Research, 10,* 459–472.

Nakhaeizadeh, G. (1995). What Daimler-Benz has learned as an industrial partner from the machine learning project Statlog. *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023, pp. 22–26). Washington, DC: Navy Center for Applied Research in Artificial Intelligence,

Narendra, K. S. (1996). Neural network for control: Theory and practice. *Proceedings of the IEEE, 84,* 1385–1406.

Nédellec, C. (Ed.) (1995). *Proceedings of IJCAI-95 Workshop on Machine Learning and Comprehensibility*.

Neri, F. (1997). *First order logic concept learning by using a distributed genetic algorithms*. Doctoral dissertation, Dipartimento di Informatica, Universita’ di Torino, Italy. Available at http://www.di.unito.it/ neri/.

Neri, F., & Saitta, L. (1996a). A study of the universal suffrage selection operator. *Evolutionary Computation*, *4*, 87–107.

Neri, F., & Saitta, L. (1996b). Genetic algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-18*, 1135–1142.

Neri, F., Saitta, L., & Tiberghien, A. (1997). Modelling physical knowledge acquisition in children with machine learning. *Proceedings of 19th Annual Conference of the Cognitive Science Society* (pp. 566–571). London, UK: Lawrence Erlbaum Associates.

Noordewier, M., Towell, G., & Shavlik, J. (1991). Training knowledge-based neural networks to recognize genes in DNA sequences. *Proceedings of NIPS* (Vol. 3). Denver, CO.

Piatetsky-Shapiro, G. (1997). Data mining and knowledge discovery: The third generation. *Proceedings of International Symposium on Methodologies for Intelligent Systems* (pp. 48–49). Berlin, Germany: Springer-Verlag.

Provost, F., & Danyluk, A. (1995). Learning from bad data. In D. Aha, & P. Riddle (Eds.), *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023, pp. 27–33). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Quinlan, R. (1993). *C4.5 Programs for Machine Learning,* San Francisco, CA: Morgan Kaufmann.

Rudenström, Å. (1995). *Applications of machine learning*. Doctoral dissertation, Department of Computer and Systems Science, University of Stockholm, Sweden (TR 95-018, ISSN 1101-8526).

Saitta, L., Botta, M., & Neri, F. (1993). Multistrategy learning and theory revision. *Machine Learning*, *11*, 153–172.

Saitta, L., Giordana, A., & Neri, F. (1995). What is the real world? *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023, pp. 34–40). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Schwabacher, M., Hirsh, H., & Ellman, T. (1995). Inductive learning from engineering design optimization. *Working notes for applying machine learning in practice: A workshop of the twelfth international machine learning conference* (Technical Report AIC-95-023, pp. 49–55). Washington, DC: Navy Center for Applied Research in Artificial Intelligence.

Silberschatz, A., & Tuzhilin, A. (1995). On subjective measures of interestingness in knowledge discovery. *Proceedings of the First Knowledge Discovery and Data Mining Conference* (pp. 50–56).

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

Slotta, J. D., & Chi, M. T. H. (1996). Understanding constraint-based processes: A precursor to conceptual change in physics. *Proceedings of Eighteenth Annual Conference of the Cognitive Science Society* (pp. 306–311). London, UK: Lawrence Erlbaum Associates.

Smith, C., Snir, J., & Grosslight, L. (1992). Using conceptual models to facilitate conceptual change: The case of weight-density differentiation. *Cognition and Instruction*, *9*, 221–283.

Tiberghien, A. (1994). Modelling as a basis for analysing teaching-learning situations. *Learning and Instruction*, *4*, 71–87.

Uberbacher, E. C., & Mural, R. J. (1991). Locating protein coding regions in human DNA sequences by a multiple sensor-neural approach. *Proceedings of National Academy of Science* (Vol. 8, pp. 11261–11265).

van Someren, M. (1994). Notes on machine learning and knowledge acquisition. *Proceedings of MLNet Summer School on Machine Learning and Knowledge Acquisition* (pp. 199–212).

Vosniadou S., & Brewer, W. F. (1994). Mental models of the day/night cycle. *Cognitive Science*, *18*, 123–183.

Widrow, B., Rumelhart, D. E., & Lehr, M. A. (1994). Neural networks: applications in industry, business, and science. *Communication of the ACM*, *37*, 93–105.

Xu, Y., Mural, R. J., & Uberbacher, E. C. (1996). Inferring gene structures in genomic sequences using pattern recognition and expressed sequence tags. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology* (pp. 344–353). Menlo Park, CA: AAAI Press.

Xu, Y., Mural, R. J., Einstein, S. M. B., & Uberbacher, E. C. (1996). GRAIL: A multi-agent neural network system for gene identification. *Proceedings of the IEEE*, *84*, 1544–1552.

Received March 4, 1997
Accepted November 15, 1997
Final Manuscript November 28, 1997