Workshop on Current Trends and Applications of Artificial Intelligence in Education, ITESM, Mexico, 1998, 98-105

Co-evolutionary Learning: Machines and Humans Schooling Together

Elizabeth Sklar
Dept. of Computer Science
Brandeis University
Waltham, MA 02254, USA
+1-781-736-2741
sklar@cs.brandeis.edu

Alan D. Blair
Dept. of Computer Science
University of Queensland
4072, Australia
+61-7-3365-1999
blair@cs.uq.edu.au

Jordan B. Pollack
Dept. of Computer Science
Brandeis University
Waltham, MA 02254, USA
+1-781-736-2741
pollack@cs.brandeis.edu

Abstract

We consider a new form of co-evolutionary learning in which human students and software tutors become partners in a cooperative learning process. While the students are learning from the tutors, the tutors will also be learning how to do their job more effectively through their interactions with the students. Earlier work on game-playing machine learners has brought to light important issues for co-evolutionary learning; in particular, that these interactions can be modeled as a *meta-game* between teachers and students, and that learning may fail due to *suboptimal equilibria* – for example, because of *collusion* between the individual players – in this meta-game of learning. We discuss some of the challenges that these issues may raise for the designers of intelligent software tutoring systems and describe a prototype Java applet that we have built in an effort to explore how these challenges may best be met.

1. Introduction

Advancing technology is opening up exciting new possibilities for education. We can envision a future paradigm in which students play a much greater role in structuring their own education, with the assistance of sophisticated software agents that can guide them through the learning process, adapting to their own individual needs. These software tutors could fulfill a number of functions – searching for information requested by the students, directing the students to appropriate sources and multimedia teaching materials, asking questions, suggesting certain topics or lines of inquiry – making sure the students gain a good grasp of the core concepts while encouraging development of a student's own particular skills and interests.

This interaction can be viewed as a form of *co-evolutionary learning* between the human students and the software tutors. While the students are learning from the tutors, the tutors will also be learning how to do their job more effectively through their interactions with the students.

We have studied co-evolutionary learning in another context – namely, that of machine learners improving their strategies in competitive games by playing each other and observing the results. Such studies have brought to light important issues for co-evolutionary learning – in particular, that these interactions may be modeled as a *meta-game* between teachers and students, and that learning may fail due to *suboptimal equilibria* – for example, because of *collusion* between the individual players – in this meta-game of learning. We believe that similar issues are likely to arise in the context of human-student/software-tutor co-evolution, and that a better understanding of these issues will help improve the design and implementation of software tutoring systems.

This paper is arranged as follows: Section 2 provides background on co-evolutionary and human learning. Section 3 discusses the problems of collusion and suboptimal

equilibria in the meta-game of learning. Section 4 examines certain game learning domains which seem to have avoided these problems. Section 5 describes an experimental educational software Java applet that attempts to address some of these issues.

2. Co-Evolutionary Learning, Human Learning

The goal of *machine learning* (ML) is to design software systems that can learn, through interacting with their environment, information which will help them to perform particular tasks better. Such systems, if successful, will automatically adapt to new environments without re-programming, thus saving humans the trouble of designing all the relevant features by hand.

The success of a machine learning system depends very much on the learning environment in which it is placed. Typically, such a system extracts information from its environment, as it is ready; then it may need to be placed in a new environment in order to progress. *Incremental* learning (Langley, 1995) occurs when a learner is placed into a pre-defined series of environments one after the other, as it progresses. However, designing an appropriate series of environments is difficult and expensive. These costs could be avoided if there were some way for the learner and its environment to *co-evolve* with each other, so that the one would always be appropriate for the other.

The subfield of co-evolutionary machine learning attempts to achieve just this, building on earlier work in genetic algorithms and genetic programming. Some pertinent applications include Prisoner's Dilemma (Axelrod, 1984; Haynes et al., 1995), the game of tag (Reynolds, 1994), tic-tac-toe (Angeline and Pollack, 1993), and backgammon (Pollack et al., 1996).

How does the notion of co-evolutionary learning apply to human learning, and, in particular, fit into today's framework of intelligent tutoring? The current trend in educational culture transfers control away from the traditional teacher in favor of the student, the learner (Corte, 1995). In this type of *constructionist* environment, students are able to explore ideas for themselves without having to stick to a fixed curriculum (Papert, 1993); and students at all levels of ability are provided with an opportunity to learn. Forrester promotes *learner-centered learning*, where students are not considered "passive receptors". Students and teachers participate in the learning process together as colleagues (Forrester, 1992).

The issue of *learner control* is addressed by John Seely Brown. He describes systems where the "locus of control" belongs either solely to the computer (e.g. frame-based CAI) or solely to the student (e.g. LOGO (Papert, 1980)). He cites disadvantages with both extremes – with the former, the student is restricted to a fixed path and cannot explore different avenues of interest; with the latter, the student may get stuck and may not be exposed to the full range of problems in the domain being learned. Brown maintains that some combination of the two schools is best, but is also the hardest to build (Brown and Burton, 1978).

Intelligent Tutoring Systems (ITS) branch out from frame-based approaches, exhibiting such desirable characteristics as providing a trace of a problem-solving session, individualizing for each user, dynamically selecting what to do next and coaching the user at opportune times (Clancey, 1986). Early work combined some or all of these, each emphasizing different issues, such as memory modelling (Schank, 1981; Kolodner, 1983), construction of rules (ACT) (?), modelling student's misconceptions (VanLehn, 1983; Soloway et al., 1981). But all of these require considerable engineering effort on the part of the system designer/programmer.

More recent work is often still highly dependent on a large engineering effort. POLA (Conati and VanLehn, 1996) uses probabilistic methods to model students solving problems in introductory physics. This work involves complex analysis of the problems being solved and attempts to combine knowledge tracing and model tracing. INSTRUCT (Mitrovic et al., 1996) tries to induce models of procedural skill by observing

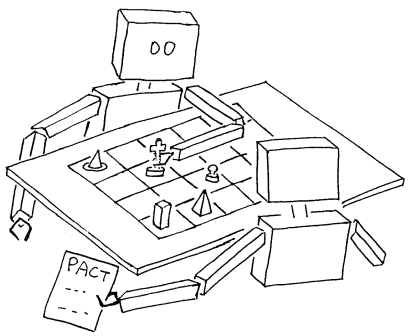
students and collecting information both implicitly and explicitly.

Intelligent Tutoring Systems take on harsh criticism from (Kinshuk and Patel, 1996), who state that "the absence of substantial and usable ITS after years of research and development activities should encourage a re-examination of the objectives and the design principles of a tutoring system." In this work, we are attempting just such a re-examination by considering a role for co-evolutionary learning in an ITS. We believe that the same environmental elements necessary for a successful co-evolutionary learning scenario are also relevant for achieving success in human learning.

3. Games, Meta-Games and Collusion

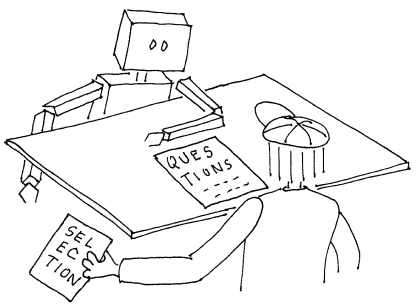
A good opportunity for studying co-evolutionary machine learning is provided by strategic games like chess, tic-tac-toe or backgammon. The theory is that several machine learners trying to master a competitive game could all learn to improve their strategies simultaneously by playing each other and observing the outcomes. Each player would be acting as a learner in a learning environment defined by its pool of potential opponents. While this idea has been around since the early days of artificial intelligence (Samuel, 1959; Michie, 1961), interestingly some applications of it have been very successful while others have run into serious difficulties.

While attempting to understand the nature of these difficulties, we have observed that a machine learning system can, in general, be modeled as an interaction between a teacher and a student (and possibly other participants) and that the manner in which these participants are rewarded defines a *meta-game of learning*, or MGL (Pollack and Blair, 1998), to which we apply a game theoretical analysis. The game-playing machine learner could be thought of as the student, while its opponent assumes the role of the teacher. The teacher's goal is to discover the student's weaknesses and help to correct them, while the student's goal is to placate the teacher and avoid further correction. In the case of game-playing learners, opposing players essentially take turns filling the roles of teacher and student and are thus evaluating *each other*. This situation provides opportunities for *collusion* between teacher and student, which show up as *suboptimal equilibria* in the MGL. Many of the problems encountered by co-evolutionary machine learners can be attributed to suboptimal equilibria of this kind.



co-evolutionary machine learners evaluate each other by playing games and observing the outcomes; they can *collude* by:

- *narrowing the scope* – playing the same kinds of games over and over
- *silent pact* – trading easy wins or playing for a 'draw', rather than a win or loss



software tutor and human student evaluate each other through a system of selection, attention and evaluation; they can *collude* by:

- *narrowing the scope* – sticking to material that is particularly favored by the student
- *silent pact* – avoiding difficult topics and providing entertainment in exchange for positive evaluation

Figure 1 The MGL: a picture of collusion

We believe the interaction between human students and software tutors can also be modeled as an MGL, as illustrated in Figure 1. This MGL differs from that of game-playing learners, taking into account the complex mix of motivations that the human student brings to the educational task. Some key issues are:

- Students should be exposed to the full *scope* of material being learned. But...
  - if the tutor is rewarded for providing more comprehensive coverage, the opportunity to be flexible and adapt to each student's individual needs may be lost; and
  - if the students are presented with too much unfamiliar material, they may become confused; i.e.

...a suboptimal equilibrium may result where the students are exposed to a broad range of material but are unable to absorb it.

- Students should receive positive *feedback*. But...
  - if the tutor is rewarded for just asking questions the student is able to answer, the tutor may stick to a narrow range of material already familiar to the student and avoid new or difficult concepts;
  - if the students are always shown familiar material, they may get bored; i.e.

...a suboptimal equilibrium may result where the scope of the material is narrowed and the students are not learning new things.

- Students should be *challenged* to explore new ideas. But...
  - if the tutor is rewarded for just asking questions the student is *unable* to answer, the tutor may end up asking questions that are well beyond the student's ability; and
  - if students are constantly attempting questions they can't answer, they may get discouraged; i.e.

...a suboptimal equilibrium may result where the students are confused and not moving ahead.

- Students should be *interested* and *attentive*. But...
  - if the tutor's reward is based just on the attention, selection and/or explicit evaluation provided by the student, the tutor may become either obsolete (where the student is self-taught) or a mere entertainer (pandering to the student's whims in ways that do not promote learning); and
  - if the students are rewarded based on their ability to 'get along' with the tutor, their own interests and needs may be subverted; i.e.

...a suboptimal equilibrium may result where the evaluation mechanism distorts the learning process.

These issues of *scope*, *feedback*, *challenge*, *interest* and *attention* need to be considered both individually and collectively, allowing for dynamical adjustment of each during the learning process. We believe that a careful balancing of these criteria will be required in order to avoid suboptimal equilibria in the MGL.

4. Avoiding Suboptimal Equilibria and Collusion

There have been a few notable cases of ML applications in which the problems of suboptimal equilibria and collusion have apparently been avoided. One such instance came to light when (Tesauro, 1992) compared two different methods for training neural networks to play the game of backgammon. The first network was trained on a large database of hand-crafted positions, with corresponding moves chosen by a human expert; the second network was trained by having it play against itself thousands of times and using the outcome of each game to make a small adjustment in its strategy according to the *temporal difference* (Sutton, 1988). Surprisingly, the network that was trained by self-play, although initially playing a poor (essentially random) game, eventually surpassed the network trained on the expert database.

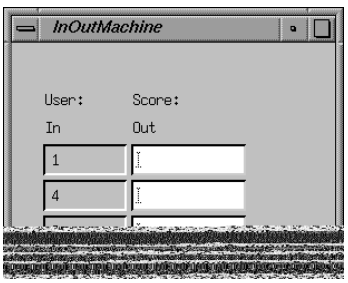
A second example is provided by the *evolving virtual creatures* (EVC) domain. In a game devised by Karl Sims, two virtual creatures compete in a world with simulated physics for control of a cube initially placed between them (Sims, 1995). In each round of competition, all creatures from one species played against the champion of the other species from the previous round. Over several generations, competing species were observed to leap-frog each other in evolutionary arms races, as they each discovered methods for reaching the cube and then further evolved strategies to counter the opponent's behaviour.

While the exact reasons for these successes were unclear at first, our hypothesis is that these two domains have special attributes which help to prevent sub-optimal equilibria in the MGL. In backgammon, the randomness of the dice rolls sometimes allows a weak player to score a victory over a strong one – an experience from which both can learn – and also tends to lead the learner into a much larger portion of the strategy space than would likely be explored in a deterministic game (Tesauro, 1992). Moreover, 'draws' are impossible in backgammon and its particular dynamics work to prevent trading of easy wins (Pollack and Blair, 1998).

Both the backgammon and EVC domains provide a broad spectrum of opportunity such that the machine learner has available to it, at any given time, a number of avenues for improvement and a number of ways to proceed along each avenue. In backgammon, there are many aspects of the game which can be developed independently. Virtual creatures, like their biological counterparts, can improve by developing a slightly longer arm, slightly better sensors, or becoming slightly faster, etc. These are in contrast to some other applications, where learning can only proceed along a set path.

Perhaps these domains providing a broad spectrum of opportunity, where learners can explore in any order, may be compared with *constructionist* learning environments that allow the user to explore a problem from a number of different angles, and thus gain a more solid grasp of the fundamental concepts (Wilensky, 1996). It would be interesting to see whether judicious attention to these attributes in the design of software tutoring systems might work to prevent collusion in the MGL.

5. The InOutMachine: A Prototype

We have built an educational software tool as a prototype for exploring these concepts. Embedded in the application are software tutors, or *tubots*. The prototype is called InOutMachine (see Figure 2), which students can use to practice simple arithmetic.

InoutMachine is based on the fundamental principle that arithmetic equations start with an input, apply a rule to that input and end with a related output. Users (students) are asked to provide the "missing elements" – the input and/or output – for a given set of addition, subtraction, or multiplication problems.