A CRCD Experience: Integrating Machine Learning Concepts into Introductory Engineering and Science Programming Courses

M. Georgiopoulos, I. Russell, J. Castro, A. Wu, M. Kysilka, R. DeMara, A.Gonzalez, E. Gelenbe, M. Mollaghasemi
University of Central Florida/University of Hartford

Abstract

Machine Learning has traditionally been a topic of research and instruction in computer science and computer engineering programs. Yet, due to its wide applicability in a variety of fields, its research use has expanded in other disciplines, such as electrical engineering, industrial engineering, civil engineering, and mechanical engineering. Currently, many undergraduate and first-year graduate students in the aforementioned fields do not have exposure to recent research trends in Machine Learning. This paper reports on a project in progress, funded by the National Science Foundation under the program *Combined Research and Curriculum Development (CRCD)*, whose goal is to remedy this shortcoming. The project involves the development of a model for the integration of Machine Learning into the undergraduate curriculum of those engineering and science disciplines mentioned above. The goal is increased exposure to Machine Learning technology for a wider range of students in science and engineering than is currently available. Our approach of integrating Machine Learning research into the curriculum involves two components. The first component is the incorporation of Machine Learning modules into the first two years of the curriculum with the goal of sparking student interest in the field. The second is the development of new upper level Machine Learning courses for advanced undergraduate students. The paper will describe the first phase of the project, that of the integration of Machine Learning concepts into introductory engineering and science programming courses through appropriately designed programming projects.

1. Introduction

Machine Learning is concerned with building computer systems that have the ability to improve their performance in a given domain through experience. In the last decade there has been an explosion of research in Machine Learning. A contributing factor in this growth is that traditionally independent research communities in symbolic Machine Learning, computational learning theory, neural networks, genetic algorithms, statistics, and pattern recognition have achieved new levels of collaboration. The outcome has been a plethora of results in Machine Learning emerging from all of these research communities working synergistically. The second reason for the explosive growth is that Machine Learning has been applied successfully to a

growing range of problems in science and engineering, such as speech recognition, handwriting recognition, medical data analysis, game playing, knowledge data discovery in databases, language processing, robot control, and others [10,8,2,13,7,14,9].

A group of faculty from the Electrical Engineering, Computer Engineering, Computer Science, and Industrial Engineering programs at the University of Central Florida have been involved in an educational program, entitled Combined Research and Curriculum Development, recently funded by NSF. The major goal of this project is to incorporate our own research results in Machine Learning, as well as the results of others, into the undergraduate engineering and science curriculum. The team of researchers that we have put together to achieve this goal comes from a variety of Engineering and Science disciplines and brings a diversity of expertise from within the Machine Learning field and its related applications. As a result, the potential for achieving our goal is high.

Furthermore, we have created a CRCD Advisory Board consisting of faculty members from various Universities around the nation, and a number of researchers from the industry and government sectors, all of whom have expertise in Machine Learning. The purpose of the Board is to assess and evaluate our effort, and at the same time, to devise effective ways of disseminating this information to the Universities affiliated with this project, as well as to other Universities. To facilitate this process of on-going feedback and evaluation as well as dissemination of material, we have planned a number of symposia throughout the duration of the project, where the results of our effort will be illustrated and feedback from the Board members will be solicited. Furthermore, more frequent feedback from CRCD members will be obtained through the project's website at http://www.seecs.ucf.edu/ml.

2. Project Overview

Our CRCD project involves a comprehensive approach to the development of a model for the integration of Machine Learning throughout the entire engineering and science curriculum. The goal is increased exposure to Machine Learning technology for a wider range of students in science and engineering than is currently available. Specifically, the overall objective of this project is to introduce, through appropriate course work, undergraduate students and first year graduate students to the recent research trends in Machine Learning. The objective will be accomplished by incorporating Machine Learning research modules into undergraduate classes and by introducing a sequence of two new Machine Learning courses, entitled Current Topics in Machine Learning, at the senior undergraduate and first year graduate levels. More specifically, our goals are:

- Incorporate current state-of-the-art Machine Learning research results into the undergraduate and first year graduate curriculum in a way that enhances the students' critical thinking, intellectual growth and communication skills.

- Offer a unique curriculum development, by traditional undergraduate standards, where faculty integrate their current research results into the curriculum. This curriculum will be

interesting and dynamic, reflecting changes in the faculty's and the Machine Learning community's research interests over time.

- Offer the opportunity to a multi-disciplinary group of students (spanning the entire Engineering spectrum of electrical, computer, industrial, civil, mechanical, as well as computer science students) to benefit from this innovative research and curriculum development.

- Assess and evaluate the impact of our efforts through a sequence of carefully chosen evaluation instruments, developed by our education specialist.

- Disseminate the curriculum development efforts to other Universities.

The students that will choose to participate in this CRCD program will go through a number of educational experiences that are outlined below. The educational experiences are broken down into four major categories: **Lectures**, **Discussions/Projects, Industry Interactions,** and **Presentations.**

Lectures:

1. *Machine Learning Modules in introductory level Engineering and Computer Science classes.*
2. *New Courses entitled Current Topics in Machine Learning I (CTML-I) and Current Topics in Machine Learning II (CTML-II).*
3. *Lecture series at the School of Electrical Engineering and Computer Science (SEECS) seminar presented by Machine Learning experts.*

Discussions/Projects:

4. *Daily interaction with the graduate students at the Machine Learning lab (during CTML-I and CTML-II courses).*
5. *Individual weekly meetings with the affiliated PI to discuss project (during CTML-II course).*
6. *Monthly group meetings with all the PIs, graduate students, and other CRCD students to report progress of work (during CTML-II course).*
7. *CRCD student discussions with the Machine Learning experts invited to participate at the SEECS seminar series.*
8. *Panel discussion with the CRCD Advisory Board at the CRCD symposium.*

Industry Interactions:

9. *Visits to industrial sites. Interaction with PIs' sponsors, and CRCD Advisory Board members.*
10. *Panel discussion with the CRCD Advisory Board at the CRCD symposium.*

Presentations:

11. *Monthly group meetings with all the PIs, graduate students, and other CRCD students to report progress of work (during the CTML-II course).*
12. *Presentation of the project results at the CRCD symposium.*
13. *Selected participation in conferences after the CRCD experience is over.*

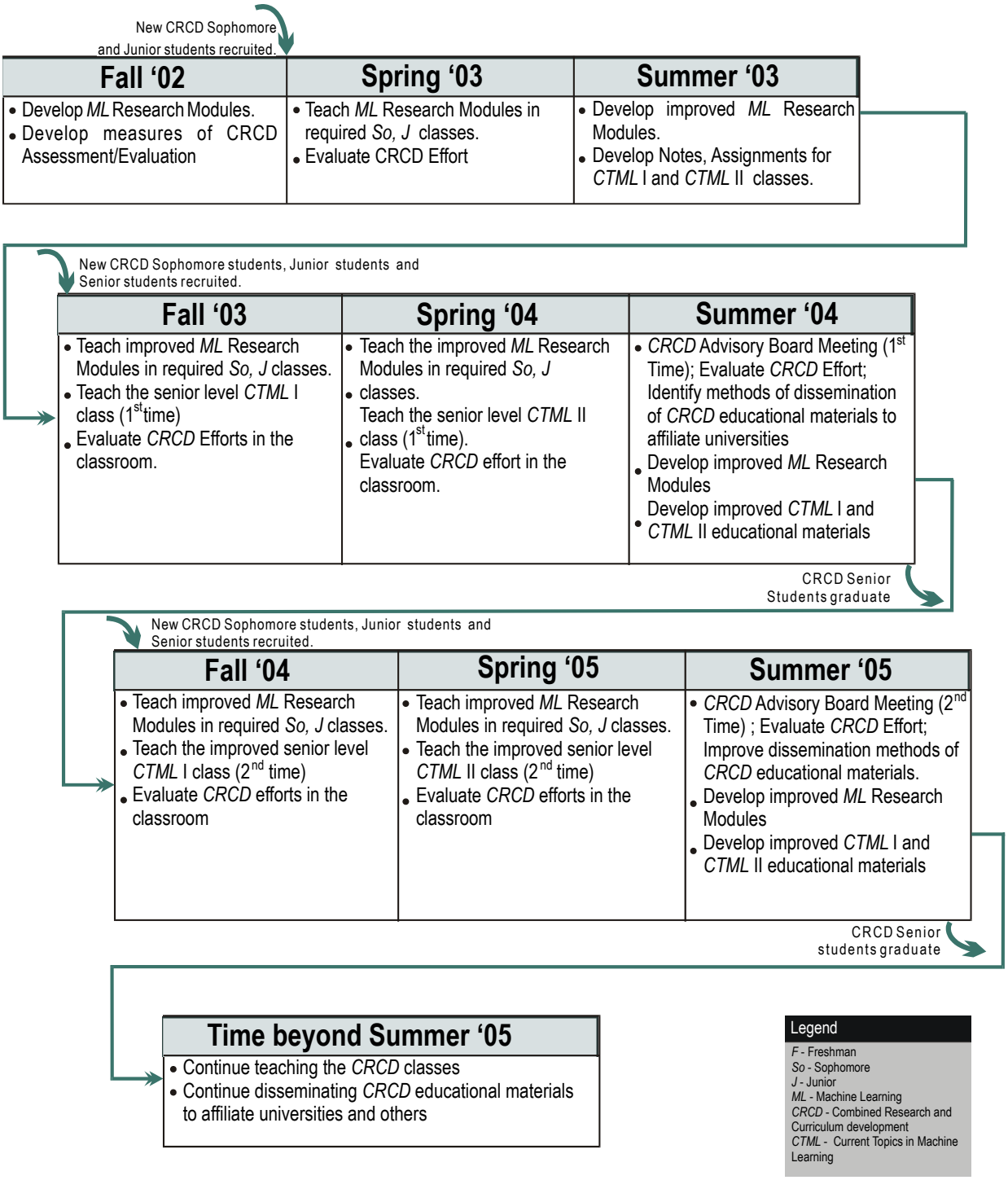

Figure 1: CRCD Timeline

Note that some of the above student activities are mentioned more than once (e.g., activities 6 and 11) because they fall into more than one category. A detailed timeline of the various educational activities that the CRCD students will get involved with are depicted in Figure 1.

The paper focuses on the first phase of the project, that of the integration of Machine Learning modules into the introductory engineering and science programming courses.

3. Machine Learning Modules

The modules that are being developed as part of this project introduce students to some of the most widely used algorithms for machine learning and their underlying principles. This is an on-going effort, and as time progresses more machine learning modules will be introduced and previous modules will be updated based on feedback from their implementations.

In the following sections, we present two Machine Learning modules that have been developed. The approach taken here is to use these Machine Learning modules as programming projects in the introductory programming course with the goal of stimulating student interest in Machine Learning while at the same time introducing various programming concepts.

The first module described in this paper uses a reinforcement learning technique. It learns to play the game of Nim by trial and error. The program starts out by playing randomly, and records the success or failure of its decisions so that in time it will be able to perform better. Games such as the Towers of Hanoi, Nim, Monkey and Bananas, and 15-puzzle can provide an effective means to demonstrate Machine Learning strategies to students. Games are easier for students to comprehend and more manageable to code than complex applications, yet retain the essential features of Artificial Intelligence (AI) mechanisms. For conveying Machine Learning concepts, the ancient Chinese game called *Nim* is ideal. By programming and tuning their own implementation of the Nim game, students can grasp and explore reinforcement learning techniques while writing a relatively short program.

The second module provides an introduction to the basic concepts of a Genetic Algorithm (GA). It consists of an interactive exercise that demonstrates the functionality of a GA. The entire class participates in creating and exploring some of the basic dynamics of how a GA works. Interactive learning methods have been found to be effective and sometimes better methods for teaching than traditional lecture-based teaching approaches[1]. Interactive exercises can provide students with a memorable, hands-on experience as well as a new point of view as they "become" a part of the topic that they are learning [3,4].

Both modules can be used and assigned to introduce the coding concepts of arrays and matrices, random numbers, function definition and invocation, and input/output loops in a computer engineering or computer science introductory programming class. The first module has already been assigned in our introductory programming course. The second will be expanded to include a programming component to implement the algorithm and will be assigned this semester.

4. The Game of Nim Project: Learning to Play by Trial and Error

4.1 Objective

The purpose of this Machine Learning module is to allow students to understand the concepts of arrays and matrices, random numbers, function definition and invocation, and input/output loops. Furthermore, students are expected through this module to attain some basic understanding of reinforcement learning concepts. A sample of student code is included in Appendix I.

4.2 Preliminaries

The field of Machine Learning is concerned with building computer systems that have the ability to improve their performance from their experience. Machine learning can be classified into three groups: supervised learning, unsupervised learning, and reinforcement learning. In a reinforcement learning model, learning is done through experience from active interaction with the environment. Unlike other learning methods, no explicit teaching is present and the actions to be taken are not given. Instead, through these experiences and interactions with the environment, the goal is to learn a procedure for selecting the optimal action taken, one that yields the most total reward from the environment [12].

An example of reinforcement learning might be a program that has to learn to play a game (checkers, for example) given only the game's rules. To achieve this, the program could use any combination of logic, trial and error, probability and any other means available. In this sense, reinforcement learning is much less constrained and general than other types of learning, and deals with the problem of an agent that has to learn to interact with its environment. It might very well be that in the process of learning, the intelligent agent would use other types of learning to extract information from particular sub-problems it has to solve.

4.3 Problem Statement

The student is to write a program that learns to play the game of Nim. There are different variations of the game, and various learning algorithms have been applied to the game of Nim [11].

In our version of Nim, there is one pile of sticks/coins/stones or any other appropriate denomination (we will call them sticks). Two players take turns in removing sticks from the pile; there is a predetermined maximum amount MAX of sticks to remove from the pile on any one turn. The player that removes the last stick loses.

This program allows the computer to play a game of Nim against a human opponent. The opponent enters the initial size of the pile of sticks, the maximum number of sticks that can be removed at a time, and whether or not the opponent wants to go first. A reinforcement learning technique will be implemented, and learning to play is accomplished by trial and error.

An example of a possible interaction with the machine is shown below (user input is in New Courier bold italic).

```
This program allows the computer to play a game of Nim against a
human opponent.  The opponent enters the initial size of the
pile of sticks, the maximum number of sticks that can be removed
at a time, and whether or not the opponent wants to go first.
The player to select the last stick loses the game.

Enter the initial size of the pile of sticks: 12
Enter the maximum number of sticks that can be removed at a
time: 3
Would you like to go first? N
The computer picks 3, there are now 9 sticks left.
Your turn, how many do you want to pick? <1,2,3> 3
You picked 3, there are now 6 sticks left.
The computer picks 1, there are now 5 sticks left.
Your turn, how many do you want to pick? <1,2,3> 3
You picked 3, there are now 2 sticks left.
The computer picks 1, there are now 1 sticks left.
Your turn, how many do you want to pick? <1,2,3> 1
You picked 1, there are 0 sticks left
You picked the last stick.  The computer wins, you lose :-)
Do you want to play another game? <Y or N> N
OK, bye
C>
```

4.4 Playing strategy

Nim is a zero-sum dominated game, which means that there is a strategy such that, if used by one of the players, the player can win regardless of the moves the opponent makes. It is not too complex to see the winning strategy for this version of Nim but the idea of this homework is to make a program that *learns* how to play by trial and error.

The program starts out by playing randomly, and it records the success or failure of its decisions so that in time it will be able to perform better. This is accomplished by keeping a matrix of integers called `move[max, amount]` where `amount` is the number of sticks remaining in the pile and `max` is the current value of MAX. The entry `move[max, amount]` records the information that we possess of what to do when we have `amount` sticks on the pile and `max` = MAX. The entry `move[max, amount]` may contain one of the following values:

- -2 : Means that we don't yet have any information on how many sticks to remove in the current situation.
- n, with $n \geq 0$. Means that if there are `amount` sticks left on the pile and the maximum allowed to remove is `max` then we can win by removing n sticks from the pile.
- -1 : Means that we can't win in this situation if the opponent plays smart, so let's just eliminate 1 piece and hope the opponent makes a mistake.

The first time that Nim is played we set `move[max,0] = 0`, and all other entries to -2. Assuming that the matrix `move` has 3 rows and n columns at the beginning of play the matrix will look like this:

`move` matrix

	0	1	2	3	4	5	6	7		n-2	n-1	n
1	0	-2	-2	-2	-2	-2	-2	-2	...	-2	-2	-2
2	0	-2	-2	-2	-2	-2	-2	-2	...	-2	-2	-2
3	0	-2	-2	-2	-2	-2	-2	-2	...	-2	-2	-2

Each time a game is played it can only alter the value of one of the rows of the matrix. Each row represents what the machine knows for one value of `max`. What the machine learns for one value of max is not generalized for other values of `max`.

Setting column 0 to zeroes says that 0 sticks on the pile is a winning position and that the winning strategy is removing 0 sticks from the pile. This is another way of saying that if you have 0 sticks on the pile then you have already won. The fact that the rest are –2's tells us that the machine does not yet know anything else.

If we have amount sticks on the pile, and the maximum allowed to be removed at a time is max, then a strategy for a move would be the following.

1. If the value of `move[max,amount] = k ≥ 0` then we know it's a winning position and how many to remove: remove k from the pile.
   That is amount = amount - k
2. If the value of `move[max,amount] = -1` then this is a losing position, so remove 1 from the pile and hope for the best (amount = amount – 1)
3. If the value of `move[max,amount] = -2` then we don't know if this is a winning or a losing position, so check the following:
   a. If there exists `i ≤ max` such that `move[max,amount-i]=-2,` then eliminate `i` sticks from the pile `(amount = amount – i).`
   b. If (a) was not successful, then label this amount as a losing position by setting `move[max, amount] = -1.` Label all positions that reach this one in a single move as winning positions. That is, set `move[max, amount+i] = i,` for all `i∈{ 1, 2, …, max}.`

For example, if we start playing with value of max of 3 and an amount of 7 (the maximum we can eliminate each turn is 3 sticks and at this moment we have 7 sticks on the pile), the row would look like this

	0	1	2	3	4	5	6	7	
3	0	-2	-2	-2	-2	-2	-2	-2	...
								^ amt	

If it is the machine's turn, it will check if the entry at index 7 is greater than 0 (point 1), since it's not it will check if it is –1 (point 2), and since it's not it can only be –2, so it will search for another –2 in the indexes of 6, 5, and 4 and decrement `amount` accordingly.

Note that learning (or array modification in 3.b of the algorithm) is only done if the machine finds a losing position from which it does not have reachable losing positions. At the start of the game this only happens for amount = 1. This triggers action 3.b of the algorithm and will change the entries of the array to

	0	1	2	3	4	5	6	7	
3	0	-1	1	2	3	-2	-2	-2	...
		^ amt							

This indicates that having 1 stick is a losing position. Having 2 sticks is a winning position and the optimum move is to remove 1 stick. Having 3 sticks is a winning position and the optimum move is to remove 2 sticks. Having 4 sticks is a winning position and the optimum move is to remove 3 sticks. When all the –2's are eliminated from the matrix the learning is complete. This will happen if we play enough games.

4.5 Program Goal and Output

Your program should be able to play the game of Nim with a human and should use the previous strategy to learn how to play. The program should check if the input is valid, if, for example, a negative number is given as the value for `max`, the program should display an error and prompt the user again for a valid value.

The program should also be able to play itself upon request from the user. This will allow the system to learn the optimum playing strategy much faster.

4.6 Discussion Issues

The module allowed discussion of various issues related to Machine Learning including:

- Program performance issues and program limitation as a function of the programmer's knowledge and various constraints.
- The importance of learning in allowing a system to expand its capabilities by adapting and learning from experience.
- Learning by trial an error and its concept of adjusting its rules based on experience to avoid the duplication of errors.
- Relationship of this to the human thought process involving knowledge based on experiences.

5. An Interactive Genetic Algorithm Project

5.1 Objective

The purpose of this module is to demonstrate the basic functionality of a genetic algorithm and explore the basic concepts using an interactive exercise. This exercise is followed by a discussion period to discuss student reactions to the exercise and to relate the exercise to the computational algorithm.

5.2 Preliminaries

A genetic algorithm (GA) is a learning algorithm based on principles from natural selection and genetic reproduction [5,6]. Key features that distinguish GAs from other search methods include:

- A population of individuals where each individual represents a potential solution to the problem to be solved.
- A fitness function which evaluates the utility of each individual as a solution.
- A selection function which selects individuals or "parents" for reproduction based on their fitness. The selection function probabilistically exploits good individuals from the current population.
- Idealized genetic operators which create new individuals from selected parents. Genetic operators attempt to explore new regions of the solution space while still retaining useful information from past solutions.

The basic steps of a GA are as follows:

```
procedure GA
   {
   initialize population;
   while termination condition not satisfied do
      {
      evaluate current population;
      select parents;
      apply genetic operators to parents to create offspring;
      current population = new offspring population;
      }
   }
```

The GA iterates through an evaluate-select-reproduce cycle until a solution is found or a user-defined stopping condition is satisfied. In the process, the GA attempts to balance the exploration for new information and the exploitation of existing information in a way that optimizes its learning process.

5.3 Activity Description

We use a simple learning method – positive reinforcement training – as our learning method. The goal of the class is to learn an action or activity that one or more trainers specify.

We begin by asking the class to select a representative to be the learner. The trainer teaches the learner to perform an action using positive reinforcement training. The rest of the class observes but cannot communicate with the learner.

We then repeat the exercise using multiple, communicating learners. There are now three trainers who decide on a single goal. The rest of the class must learn this goal. Each trainer is assigned one initial learner for a total of three learners. The activity then proceeds as follows:

1. Trainers use positive reinforcement training to teach their respective learners for a fixed period of time.
2. If goal achieved, end activity.
3. Trainers confer to decide which learner to eliminate. Should eliminate the worst one.
4. A new learner from the class replaces the eliminated learner.
5. Learners have a fixed period of time to confer.
6. Go to step 1.

This activity simulates the basic workings of a GA. The goal represents the problem that a GA is applied to solve. Each learner represents an individual in a GA population. Communication between learners simulates exchange of information via genetic operators. The trainers who are all training for the same goal represent the fitness function. The use of a simple positive reinforcement feedback to provide "fitness" information is meant to express the fact that the feedback in a Machine Learning algorithm is often not very complex, and simple feedback can be enough to support the learning of fairly sophisticated tasks. In effect, we are running a "human GA" with a population of size three with offspring represented by the replacement members of the class.

5.4 Discussion Issues

Following the interactive exercise, we hold a discussion period in which student reactions are expressed. Discussion issues will focus on the impact of the distributed nature of the activity (as compared with the lone learner). Questions include:

- Who was in control of the learning process? Was it organized?
- How did the class feel when watching the lone and group learning exercises?
- Which approach appeared to be more efficient, more flexible, more organized?
- Is communication important? Between whom? Was there sufficient communication?

In addition to the discussion, we present a basic introduction to the computational algorithm itself and make the connection between the exercise and a GA.

6. Discussion/Experiences

Of the two Machine Learning modules presented, we have assigned and done an initial evaluation of the first one, the Game of Nim module. This module was assigned as one of the programming assignments in an introductory programming course in Computer Engineering taught to sophomore students. Although our experience with the modules and our assessment of

the effectiveness of our approach is limited, the results of our initial evaluation have been very positive. After the students completed the assignment, they were asked to evaluate their experience by completing a questionnaire. The questionnaire included a variety of questions that addressed the effectiveness of the module in practicing the programming concepts it intended to cover as well as in providing a gentle introduction to Machine Learning concepts. In addition, the questionnaire addressed students' interest in learning more about Machine Learning.

While the sample data is small at this time, student feedback from this initial use of the module was very positive. The module was well received by most students as was evidenced in the results of the questionnaire. One of the obvious conclusions from the assessment results is that using computer games as Machine Learning modules to help students learn important concepts was appealing to many students and contributed to a high self-motivation level. Despite the challenges involved in learning new concepts, the project was favorably received by students, who felt that it was conducive to learning. When asked to comment on the effectiveness of the module in introducing Machine Learning concepts, results of the survey showed 85% of the students gave a 3-5 ranking on a scale of 1 to 5 with 5 representing "a lot". When asked if they would be interested in learning more about Machine Learning, 55% of the students gave "interested" or "very interested" responses. In addition, when asked about the effectiveness of this module in exercising the programming skills and concepts of this project, 80% of the students replied with "useful" or "very useful".

Such survey results showed that students seemed to have a better appreciation of this area of research and an interest in learning more about it. At the same time, it seemed that this introduction to new concepts in the context of a programming project did not distract from the coverage and their understanding of the more traditional topics of the course.

The Genetic Algorithms module is being expanded to include a programming component to implement and code the algorithm. The module will be assigned for the first time during this semester. We will be using these and additional modules again in future semesters. We have recently completed a more thorough assessment and evaluation instrument that we plan to use in the future, a copy of which is included in Appendix II. The intent of this form is to provide us with an evaluation of the effectiveness of our modules in achieving the various objectives of the project as it relates to Machine Learning, while at the same time maintaining the goals of understanding the traditional topics covered in such a course. As such, the questions on the form are divided into two parts, one for assessing each of the two objectives just mentioned. In addition, it is our hope that this assessment instrument will help provide us with feedback as we continue to improve on and add to our suite of modules.

Conclusion

We presented an overview of and our experiences with a project in progress whose goal is to integrate research results in Machine Learning into the undergraduate and first year graduate engineering and science curriculum. In this paper, we focus on one of the instruments that we have proposed in an effort to achieve this goal. This approach relies on incorporating simple but

effective Machine Learning modules in appropriate introductory computer programming classes of the engineering and computer science curriculum.

The Machine Learning module we presented was well received by students as was evidenced by results of a questionnaire that was distributed after the completion of the module assignment. Additional modules will be introduced. It is our hope that by increasing the number of Machine Learning module assignments in these and similar undergraduate classes, we will motivate some of these students to pursue research in Machine Learning. The avenue to accomplish that is for these students to register for our planned Current Topics in Machine Learning classes, where they will have the opportunity to get involved in Machine Learning research projects that our CRCD faculty are interested in.

References

1. Beichner, R. J., Saul, J. M., Allain, R. J., Deardorff, A. L., and Abbott, D. S., "Introduction to SCALE-UP: Student-centered activities for large enrollment university physics", In *Proceedings of the 2000 Annual Meeting of the American Society for Engineering Education,* 2000.

2. Carpenter, G. A., Markuzon, N., "ARTMAP-IC and Medical Diagnosis: Instance Counting and Inconsistent Cases," *Neural Networks*, Vol. 11, No. 2, March 1998, pp. 323-336.

3. Colella, V., "Participatory simulations: Building collaborative understanding through immersive dynamic modeling," *Journal of the Learning Sciences*, 9(4), 471-500, 2000.

4. Colella, V., Klopfer, E., and Resnick, M., (2001). *Adventures in Modeling: Exploring Complex Dynamic Systems with StarLogo*. Teachers College Press, 2001.

5. Goldberg D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

6. Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

7. Kohonen, T., "Self-organizing maps of massive document collections," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Vol. 2, pp. 3-9, 2000.

8. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems*, Vol. 2, pp. 396-404, San Mateo:CA: Morgan Kauffman, 1990.

9. Pomerleau, D. A., "Neural networks for intelligent vehicles," *Proceedings of the 1992 Intelligent Vehicles Symposium*, Jun 29 – July 1, 1992, pp. 391-396.

10. Sejnowski, T. J., and Rosenberg, C. R., "Parallel networks that learn to pronounce English text," *Complex Systems*, Vol. 1, pp. 145-168.

11. Shapiro, S. C., *NimLearn: A Learning Nim Player,* http://www.cs.buffalo.edu/~shapiro/Courses/CSE572/nimlearn.ps

12. Sutton R. S., and Barto, A. G., *Reinforcement Learning: An Introduction,* MIT Press, Cambridge, MA, 1998.

13. Tesauro, G., "TD-Gammon, A self-playing Backgammon program achieves master play," *Neural Computation*, Vol. 6, pp. 215-219, 1994.

14. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J., "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-37, pp. 328-339, 1989.

Acknowledgement

The authors from the University of Central Florida acknowledge the partial support from NSF through a CRCD grant number 0203446 entitled "Machine Learning Advances for Engineering Education".

Biographical Information

MICHAEL GEORGIOPOULOS is a Professor of the School of Electrical Engineering and Computer Science at the University of Central Florida. His research interests lie in the areas of neural networks and applications of neural networks in pattern recognition, image processing, smart antennas and data-mining. He is an Associate Editor of the IEEE Transactions on Neural Networks since 2001.

INGRID RUSSELL is a Professor of Computer Science at the University of Hartford. Her research interests are in the areas of artificial neural networks, pattern recognition, semantic web technologies, and computer science education. She has been involved in several computer science curriculum projects. Most recently she chaired the Intelligent Systems focus group of the IEEE-CS/ACM Task Force on Computing Curricula 2001.

JOSE CASTRO is currently a Ph.D. student in the computer Engineering program of the University of Central Florida. His current area of interest is neural networks, parallel algorithms and data-mining. His Ph.D. topic is "Modification of the ARTMAP algorithm for efficient parallel processing on large data-sets.

ANNIE WU is an Assistant Professor at the School of Electrical Engineering and Computer Science at the University of Central Florida. Her research interests are in the areas of genetic algorithms, machine learning, biological modeling, and visualization.

RONALD DEMARA is an Associate Professor at the School of Electrical Engineering and Computer Science at the University of Central Florida. He has been a reviewer for National Science Foundation, Journal of Parallel and Distributed Computing, IEEE Transactions on Parallel and Distributed Computing. His interests lie in the areas of Parallel and distributed processing, self-timed architectures.

AVELINO GONZALEZ is a Professor of the School of Electrical Engineering and Computer Science at the University of Central Florida. He has co-authored a book entitled, "The Engineering of Knowledge-Based Systems: Theory and Practice". His research interests lie in the areas of artificial intelligence, context based behavior and representation, temporal reasoning, intelligent diagnostics and expert systems.

MARCELLA KYSILKA is a Professor and Assistant Chair of the Education Foundations Department at the University of Central Florida. She is active in her professional organizations and currently serves as Associate Editor of the "Journal of Curriculum and Supervision" (the scholarly journal of the Association for Supervision and Curriculum Development). Her research interests are in curriculum studies.

EROL GELENBE is a Professor and Director of the School of Electrical Engineering and Computer Science and an Associate Dean of the College of Engineering and Computer Science at the University of Central Florida. He is a Fellow of IEEE and a Fellow of ACM. His research interests cover packet network design, computer performance analysis, artificial neural networks and simulation with enhanced reality.

MANSOOREH MOLLAGHASEMI is an Associate Professor at the Industrial Engineering and Management Sciences (IEMS) Department at the University of Central Florida. She has co-authored three books in the area of Multiple Objective Decision Making. Her research interests lie in Simulation Modeling and Analysis, Optimization, Multiple Criteria Decision Making, Neural Networks and Scheduling.

Appendix I

Student Sample Code for the Game of Nim Module

```
//include necessary libraries
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

//declare global variables
int amountStart;

//declare function prototypes
void initMove(int, int, int[5][31]);
int compMove(int, int, int[5][31]);
int playerMove(int, int);
void isLosing(int *);
void isWinning(int *, int);
void isUnknown(int, int *, int[5][31]);

//define main function
void main()
{
   //declare local variables
   int max, amountLeft, turn=0, gameType=0,i,j;
   char play, goFirst, nothing, cont;

   //declare and initialize a two dimensional array
   int move[5][31]={0};

   //give brief instructions
   printf("You are about to play NIM!\n");
   printf("Whoever takes the last piece loses.\n\n");

   //start game loop
   do
   {
      //ask user for max number of pieces that can be taken
             printf("Enter MAX number of pieces that can be taken (between 1 and 5): ");
      scanf("%d", &max);
      scanf("%c", &nothing);

      //ask user for the amount of pieces to start with
       printf("\nEnter the starting amount of pieces (between 1 and 30):    ");
      scanf("%d", &amountStart);
      scanf("%c", &nothing);
      amountLeft = amountStart;

      //call function to initialize array for the current game
      initMove(max, amountStart, move);

      //ask user if he/she would like to play
      //if no, the computer will play itself
      printf("\nWould you like to play (Y = Yes, N = No)? ");
      scanf("%c", &play);
      scanf("%c", &nothing);

      //check that user entered correct value
      while(play != 'y' && play != 'Y' && play != 'n' && play != 'N')
      {
         printf("\nERROR: That is an incorrect value!\n\n");
         printf("Would you like to play (Y = Yes, N = No)? ");
```

```
   scanf("%c", &play);
   scanf("%c", &nothing);
}

//if the user wants to play, ask if he/she wants to go first
 //if/esle statement also determines the game type, player/computer    or
 computer/computer
if(play == 'Y' || play == 'y')
{
   gameType = 1;
   printf("\nWould you like to go first (Y = Yes, N = No)? ");
   scanf("%c", &goFirst);
   scanf("%c", &nothing);

   //check that user entered correct value
        while(goFirst != 'y' && goFirst != 'Y' && goFirst != 'n' &&    goFirst !=
        'N')
   {
      printf("\nERROR: That is an incorrect value!\n\n");
      printf("Would you like to go first (Y = Yes, N = No)? ");
      scanf("%c", &goFirst);
      scanf("%c", &nothing);
   }

   //determine starting turn from users response
   //turn = 0, computer / turn = 1, player
   if(goFirst == 'Y' || goFirst == 'y') turn = 1;

               else turn = 0;
}

else gameType = 0;

//determine the game type desired
switch(gameType)
{
   //computer play only
   case 0:
      //play game while the pile is not empty
      while(amountLeft > 0)
      {
         //call compMove function to get the computer's move
         //the function will return the new amount on the pile
         amountLeft = compMove(max, amountLeft, move);
      }

      //computer always wins
      printf("\nI win!\n");
      break;

   //computer and player play
   case 1:
      //play game while the pile is not empty
      while(amountLeft > 0)
      {
      //determine which player function to call from turn variable
         switch(turn)
         {
            //computer's turn
            case 0:
             //call compMove function to get the computer's move
             //the function will return the new amount on the pile
               amountLeft = compMove(max, amountLeft, move);
               break;
```

```
                  //player's turn
                  case 1:
                   //call playerMove function to get the user's move
                   //the function will return the new amount on the pile
                     amountLeft = playerMove(max, amountLeft);
                     break;
               }

               //switch turns
               if(turn == 1) turn = 0;

                                 else turn = 1;
            }

            //the winner is the next player that would have gone
            //after turn has been changed, turn holds the winner's value
            //declare the winner
            if(turn == 0)
            {
               printf("\nI win!\n");
            }

            else  printf("\nYou win!\n");
            break;
      }

      //ask user if he/she would like to play again
      printf("\nWould you like to play again (Y = Yes, N = No)? ");
      scanf("%c", &cont);
      scanf("%c", &nothing);

      //check that the user entered a correct value
      while(cont != 'y' && cont != 'Y' && cont != 'n' && cont != 'N')
      {
         printf("\nERROR: That is an incorrect value!\n\n");
         printf("Would you like to play again (Y = Yes, N = No)? ");
         scanf("%c", &cont);
         scanf("%c", &nothing);
      }

   //run the game while the user does not enter No to play again
   }while(cont != 'N' && cont != 'n');

   printf("\n\nBye!\n");
}

//define initMove function
//this function initializes the 2 dimensional array it is passed
//it also takes the max value and starting amount entered by the user
void initMove(int max, int amount, int move[5][31])
{
   //declare local variables
         int i;

      //initialize the first value in the array to 0
      move[max - 1][0] = 0;

      //set all the other values to -2 for unknown moves
      for(i = 1; i <= (amount); i++)
      {
       //only initialize values that have not been altered during a game
                  if(move[max - 1][i] == 0)
                  {
```

```
            move[max - 1][i] = -2;
                    }
      }
}

//define playerMove function
//this function gets a move from the user and returns the new amount on the pile
int playerMove(int max, int amountLeft)
{
   //declare local variables
   int take;
   char nothing;

   //ask user for the number of pieces he/she wishes to take
   printf("\nPieces left: %d\n", amountLeft);
   printf("How many would you like to take (MAX = %d)? ", max);
   scanf("%d", &take);
   scanf("%c", &nothing);

   //check that the number entered is between 1 and max
   while(take > max || take < 1 || take > amountLeft)
   {
      printf("\nERROR: You have entered an incorrect value!\n");
      printf("How many would you like to take (MAX = %d)? ", max);
      scanf("%d", &take);
           scanf("%c", &nothing);
   }

   //calculate the new amount of pieces
   amountLeft = amountLeft - take;

   //return the new amount
   return amountLeft;
}

//define compMove function
//this function calls 3 additional functions to determine the computer's move
int compMove(int max, int amountLeft, int move[5][31])
{
   //declare local variables
   int aMove;

   printf("\nPieces left: %d\n", amountLeft);

//get the value from the current array position using max and amountLeft
   aMove = move[max - 1][amountLeft];

   //determine which function to call from the value of aMove
   switch(aMove)
   {
      //if the value is -1, the computer is in a losing position
      //call isLosing function
      case -1:
         isLosing(&amountLeft);
         break;

      //if value is -2, the move is unknown
      //call isUnknown function
      case -2:
         isUnknown(max, &amountLeft, move);
         break;

      //for all other values call the isWinning function
      default:
```

```
         isWinning(&amountLeft, aMove);
         break;
   }

   //return the new amount of pieces to the main function
   return amountLeft;
}

//define isLosing function
//this function simply removes one piece from the pile as the computer's move
void isLosing(int *amountLeft)
{
   *amountLeft = *amountLeft - 1;
   printf("\nComputer took: 1\n\n");
}

//define isWinning function
//this function removes from the pile the number of pieces specified by
//the current array location, this number was determined in the compMove
//function and passed as a parameter to the take variable
void isWinning(int *amountLeft, int take)
{
   *amountLeft = *amountLeft - take;
   printf("\nComputer took: %d\n\n", take);
}

//define isUnknown function
//this function is called if the value of the current array position is -2
//the function determines whether the current position is a losing position,
//a winning position, or still unknown
void isUnknown(int max, int *amountLeft, int move[5][31])
{
   //declare local variables
   int take = 0, n, moveFound = 0, movePosition;

   //if the next position in the array holds a -2, the current move is
   //still unknown, so take a random number of pieces from the pile
   // within, 1 and max the rand function is seeded by the srand
   //function using the current time
      if(move[max - 1][*amountLeft - 1] == -2 && *amountLeft >= max + 1)
      {
         srand(time(0));
                  take = 1 + rand () % max;
      }

   //if the next position in the array is -2 and the amount left is less
   //than 1 + max, take only one piece.  The computer will most
   // certainly lose and will learn the winning move for the next game

   else if(move[max - 1][*amountLeft - 1] == -2 && *amountLeft < max+ 1)
      {
         take = 1;
      }

   //if the computer still has no move at this point, the next value in
   // the array is not -2.  This means the current position is a losing
   // one.  So, set the current array position equal to -1 for losing,
   // and all previous positions within max from this point equal to the
   // number of pieces each position is from the current position, the
   // losing one
      else
      {
         move[max - 1][*amountLeft] = -1;
```

```
      for(n = *amountLeft + 1; n <= *amountLeft + max; n++)
      {
         move[max - 1][n] = (n - *amountLeft);
      }
      take = 1;
   }

 //display the number of pieces taken by the computer and subtract
 // that number of pieces from the pile
      printf("Computer took: %d\n", take);
      *amountLeft = *amountLeft - take;
```

Appendix II

Assessment and Evaluation Form for the Game of Nim Module

Purpose

The purpose of this machine learning module is to provide students with an understanding of the concepts of arrays and matrices, random numbers, function definition and invocation, and input/output loops. Furthermore, the students are expected through this module to attain some basic understanding related to reinforcement learning techniques.

Learner Outcomes

1. Understand the concepts of arrays and matrices.
2. Understand the concept of random numbers.
3. Understand the concept of function definition and invocation.
4. Understand the concept of input/output loops.
5. Understand the basic idea behind reinforcement learning.
6. Have some understanding of where reinforcement learning can be applied.

Activity

How hard did you find the homework? Comments:	Very Hard	Hard	Normal	Easy	Very Easy
Did you enjoy doing the homework? Comments	A lot	Yes	A bit	No	Disliked it
Was the assignment useful to exercise your programming abilities? Comments:	Very Useful	Useful	A bit Useful	Not Useful	No, it was detrimental
On a scale of 1 to 5. 5 meaning a lot, 1 not at all, how much do you think the homework related you to machine learning?	5	4	3	2	1

Comments:					
Would you be interested in knowing more about machine learning?	Very Interested	Interested	Just Curious	Not Interested	Never Again
Comments:					

Evaluation, related to Learner Outcomes

1. Did you understand the concept of arrays and matrices? If you found it difficult to acquire this concept, what was the reason? Please explain.
2. Did you understand the concept of random numbers? If you found it difficult to acquire this concept, what was the reason? Please explain.
3. Did you understand the concept of function definition and invocation? If you found it difficult to acquire this concept, what was the reason? Please explain.
4. Did you understand the concept of input/output loops? If you found it difficult to acquire this concept, what was the reason? Please explain.
5. Did you understand the idea of reinforcement learning? If you found it difficult to understand it, what was the reason? Please explain.
6. Did you acquire some appreciation of where reinforcement learning can be applied? If you found it difficult to acquire this appreciation, what was the reason? Please explain.

Demographics

Gender: Male [] Female []
Level: Sophomore [] Junior [] Senior []

Have you had any previous programming experience recently? Please explain.

What is the length of the longest program that you wrote? In what programming language?