Active Learning of Introductory Machine Learning

Maja Pantic[1] and Reinier Zwitserloot[2]

Abstract - **This paper describes a computer-based training program for active learning of Agent Technology, Expert Systems, Neural Networks and Case-Based Reasoning by undergraduate students using a simple agent framework. While many Machine Learning (ML) and Artificial Intelligence (AI) courses teach ML and AI concepts by means of programming assignments, these assignments have usually no connection to how the student will apply the newly obtained knowledge to previously unseen, real-world problems. The pedagogy that we adopted here is computer-based active learning: teams of students are presented with well-defined assignments aimed at building intelligent agents for person identification and recognition of facial expressions and emotions from video recordings of their faces. Classroom experience indicates that the students found the specified programming assignments highly motivating. Objective evaluation studies suggest that students learn much more effectively when a contextualized, collaborative, constructive, and reflective approach is used than when an orthodox, objectivist approach to teaching ML and AI techniques is used alone.**

Index Terms – active learning, agent technology, machine learning, face technology.

INTRODUCTION

The human face is used to distinguish one person from another, to interpret what has been said by lip reading, and to communicate and understand somebody's affective states and intentions on the basis of shown facial expressions. Machine understanding of human facial behavior could revolutionize human-computer interaction technologies and fields as diverse as security, communication, and education. This wide range of principle driving applications has caused a surge of interest in the face technology and its integration into cars, personal computers, and small mobile devices like cell-phones and PDA devices. The availability of face technology will become increasingly ubiquitous just as most mobile phones have built-in digital cameras nowadays.

The hype about multi-agent systems is another fact. Since 1995, many of the IEEE and ACM journals have devoted special issues to software agents. Since 1997, many related conferences have taken place. Nowadays, special panels on intelligent agents are organized during industry symposia. Agents that sort e-mails, retrieve and parse information from Web pages, and represent personal assistants having a 'personality' are commercially available to date.

Although many courses teach basic ML and AI concepts such as rule-based reasoning and neural networks by means of programming assignments, these learning tasks are not usually situated in some meaningful real-world task, not to mention new-media-contextualized tasks that would introduce students to the relatively novel concepts concerning face technology and multi-agent systems. The computer-based training (CBT) program presented in this paper departs from the orthodox objectivist approach to teaching programming to novices, where assessment exercises have no real connection to how the student will apply the newly obtained knowledge and skills to previously unseen, real-world problems. It represents an active-learning program, which epitomizes learning that is interactive, student-centered, exploratory, contextualized, intentional, reflective and collaborative [2]. However, the presented CBT program does not typify a pure constructivist approach, in which students are given a vaguely specified problem statement that they should refine and for which they should then develop a solution. This decision was inspired by recent studies that have shown that programming novices, representing the target audience in our case, are not ready for such a pure constructivist teaching approach [6]. The approach that we chose was to present teams of students with a sequence of well-defined project assignments of gradually increasing complexity, and to provide them with a team-tailored (more or less worked-out) plan for realizing each of the assignments, but to allow them to build personalized versions of the pertinent project assignments. Representing both a synthesis of the objectivist and the constructivist approach to teaching [5] and a straightforward implementation of active learning as defined by Jonassen [2], this approach promised positive learning outcomes by bringing intellectual rigor while being engaging for the students.

IN2420 Machine Learning is an introductory course for second-year undergraduate computer science (CS) and electrical engineering (EE) students. By the end of this course, students should be familiar with some of the foundations of the AI and ML, to have an understanding of the basic AI concepts and ML techniques including Intelligent Agents, Expert Systems, Neural Networks, Instance Based Learning, Fuzzy Logic, and Genetic Algorithms and to gain programming skills using the Java programming language with an emphasis on ML and intelligent multi-agent systems (MAS). Enrolment is typically 75 to 95 per class. The curriculum schedules 20 class meetings of one hour each (lectures) and 60 hours of coursework (project) per student. Assessment is based on the project work and a final exam. The project is designed to build on lectures by teaching students

[1] Maja Pantic, Imperial College London, Computing Department, London, U.K., m.pantic@imperial.ac.uk
[2] Reinier Zwitserloot, Delft University of Technology, EEMCS, Delft, The Netherlands, reinierz@gmail.com

1-4244-0257-3/06/$20.00 © 2006 IEEE **October 28 – 31, 2006, San Diego, CA**

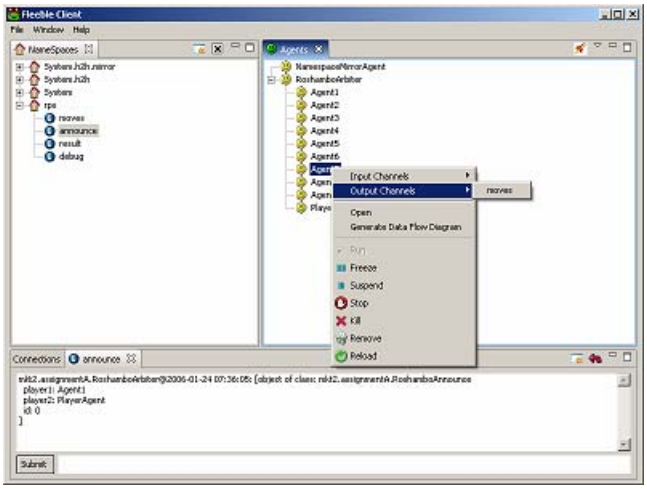

FIGURE 1
SCREENSHOT OF THE GRAPHICAL USER INTERFACE OF THE FLEEBLE.

how to create intelligent MAS using ML techniques about which they have been lectured. The project consists of three assignments: one focusing on rule-based reasoning and facial muscle action detection, one focusing on basic emotion recognition using neural networks and case-based reasoning, and one focusing on mobile agents and person identification.

The paper will begin by explaining the simple agent framework on which the developed CBT program was built. The assignments forming the CBT program in question will be described next. Then we will discuss learning effectiveness that was assessed by objective (test grades) and subjective (perceived satisfaction) measures of student learning. A discussion of the lessons learned will conclude the paper.

SIMPLE AGENT FRAMEWORK

The simple agent framework, which we utilize as the basis of the CBT program for teaching ML and AI programming to novices, is the Java-implemented Fleeble agent framework [7], [8]. Fleeble supports the development of MAS applications and embodies the concepts of concurrency, multi-agency, persistency, distribution, and mobility. In contrast to the agent frameworks developed elsewhere [12], [16], Fleeble is a suitable tool for teaching AI programming to novices because it is simple, it is accompanied by online accessible documentation, it provides readily available examples, and it embodies simple, self-explained, visual interaction with the user that may concern one, more, or all currently running agents (Figure 1) and it facilitates online generation of data flow diagrams (Figure 2).

Fleeble can be seen as a common programming interface delimiting the behavior of all the agents integrated into the framework. Its functional specification can be summarized as follows (for more information, see [7], [8]).

- Fleeble instantiates and configures the agents and start them up in separate threads. It keeps track of all agents and their parent agents. Loading one or more MAS results in one or more easily traversable trees that clearly indicate which agents are working together to accomplish a single task (Figure 1).
- Fleeble offers a message delivery system for interaction between agents that is based on a Publish/Subscribe system and centered on the concept of *channels*. A channel is a named entity that allows a single message to be delivered to any number of agents subscribed to that channel.
- Fleeble supports modular design. Students are provided with a number of building blocks including a few simple agents and a number of Java-implemented ML algorithms such as the forward chaining inference procedure.
- Fleeble supports the concept of concurrency which allows agents to operate independently and yet at the same time.
- Fleeble supports the concepts of data and state persistence. This allows the programmer to shut down a single agent (or even the entire framework) and to restore it from the point where it was suspended later on, even when the computer has been shut down in the meantime.
- Any item of an agent tree (a MAS) can be suspended, reawakened, or even destroyed. This enables the user to manually find and destroy any malfunctioning agent.
- Fleeble supports object-based communication: any object can be sent via the delivery system without the need of being converted to a text string before it can be sent.
- Fleeble supports the concepts of distribution and mobility: agents residing on different frameworks can exchange messages or can physically move to another host.
- Fleeble enables the user to emulate multiple frameworks; it offers the ability to encapsulate all channel communication for any agent such that the agent "gets the impression" that it has moved to a different host.
- Fleeble supports auto-compilation; it detects changes in the source code and automatically recompiles all files that have been changed, even if a multi-agent application is still running.
- Fleeble has a simple, easy to understand GUI. It visualizes the overall traffic through channels, the agent hierarchy, and the agents themselves. Each agent is represented by an appropriate icon suggesting the state of the agent – a green smiley for an active agent, a red smiley for an inactive agent, and a blue one for a suspended agent. By clicking the right mouse button on an appropriate entry, the user opens the relevant context menu with the related actions.
- Fleeble GUI supports monitoring of external connections**.** It makes an active network connection when it is about to transfer a message or to mobilize an agent to a remote host.
- Fleeble GUI runs independently of the main agent framework and it does not need to be started in the same physical computing environment in which the agent framework runs. A user can log into a Fleeble server either locally or remotely, via a network connection.
- The visualization of the information flow within a MAS in the form of a Data Flow Diagram (DFD) is a useful tool when designing large MAS. Such visualization can enable quick and easy estimation of whether the control flow within the MAS has been designed correctly. Fleeble

enables creation of a DFD for an arbitrary number of agents while the MAS in question runs. Agents and Channels are drawn as nodes of a network representing the MAS (Figure 2). If an agent is subscribed to a channel there is a connection from the channel to the agent. If an agent publishes to a channel there is a connection from the agent to the channel. At first, all Agents and Channels are assigned random places in the network. Then, the network is iteratively optimized to reduce the length of the connections to an a-priori defined minimum. A random amount of energy is added to ensure that the nodes can move around in the network. Manual dragging and dropping of the Agents and Channels of the network is also possible.

- To make the software installation process as easy as possible, Fleeble is distributed as an *installer* program. Finding an appropriate directory where Fleeble can be installed, setting up 'shortcuts' to Fleeble, and checking whether the appropriate version of Java Development Kit is available, are all done automatically.
- Starting Fleeble will also check if a new version of it is available. Checking Fleeble's web site (*www.Fleeble.net*) does this. Installing a newer version of Fleeble over an existing installation does not modify either preferences or stored agent persistence data.

Fleeble has been already successfully utilized for teaching basic AI programming concerning expert systems, semantic networks, search algorithms, and ad-hoc networks to first-year undergraduate CS students [7], [8]. Students perceived Fleeble as affable and suitable for the goals of teaching basic AI programming to novices. The reason is that they perceived it as useful, easy to use, and affectively qualitative. The perceived usefulness and the perceived ease of use are directly related to the above-mentioned properties of Fleeble. The perceived affective quality of Fleeble is directly tied to aesthetics qualities of Fleeble GUI [14]: it has an orderly and clear design in accordance to multiple rules advocated by usability experts and it reveals designers' creativity and originality manifested by, for example, enabling the user to interactively generate data-flow diagrams.

Because successful educational tools/interventions are not usually ubiquitously applicable and are often not readily transportable to novel settings, it is extremely important to consider generalization issues. Although Fleeble seemed to provide a basis for the development of other CS and EE courses having intelligent agents programmed to perform tasks within any specific subject area, we wanted to prove this by utilizing Fleeble as a basis for the development of an entirely different course than the one for which Fleeble has been originally developed [7], [8]. In this paper we present our efforts to utilize Fleeble for the development of the project belonging to IN2420 Machine Learning introductory course taught at Delft University of Technology to 2nd-year undergraduate CS and EE students.

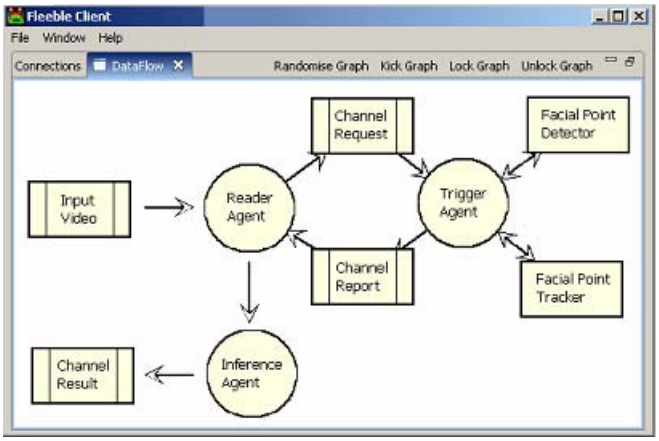

FIGURE 2
FLEEBLE-GENERATED DATA FLOW DIAGRAM DEPICTING A GENERAL STRUCTURE OF MAS TYPICALLY BUILT FOR THE 1ST ASSIGNMENT.

CONTENT OF THE EDUCATIONAL PROJECT

IN2420 Machine Learning project consists of 3 programming assignments that delve into the issues of how to employ rule-base reasoning, neural networks, case-based reasoning, and mobile agent concepts to recognize facial expressions and emotions from face videos and to retrieve relevant information from a remote Facial Expression database. The assignments have been designed for teams of 5-6 students.

While the universal aim of all the teams is to complete the project, teams of more skilled students (as measured by the speed at which they finalize the deliverables required by the assignments) are expected to deliver better end products and outcomes of the project while teams of less skilled students are provided with additional support and help. More specifically, teams of less skilled students are given chunks of code in order to alleviate "hard-core" programming tasks and to free their time for efforts in understanding ML techniques. In this way *learner-centricity* is implemented, in which the project design is adapted to the skill level of any one team of students.

I. Rule-based Facial Action Detection

The first assignment is as follows: *Create an agent-based system that detects 10 different facial muscle actions based on the spatial displacements of 20 facial characteristic points that are tracked in the input video of the subject's face by the available facial point tracker. Explain the choices made.*

Participating teams of students usually tackled this problem in the following way (Figure 2):

1. Build a *Reader Agent* that for an input frontal-face video calls a *Trigger Agent*, which executes the C-implemented versions of the facial point detector and the particle-filtering-based point tracker described in [9], and then stores per frame of the input video the results of tracking 20 facial characteristic points (Figure 4) in a text file.
2. Build an *Inference Agent* that applies forward chaining to determine which facial muscle actions (i.e., Action Units, AUs, as proposed in [2]) are displayed in the input video.

Begin by developing a knowledge base containing a set of

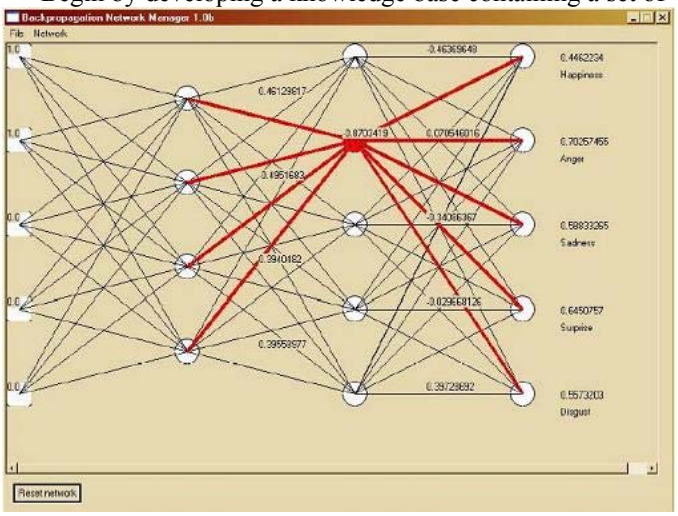

FIGURE 3
THE VISUALIZATION OF THE NEURAL NETWORK TRAINING PROCESS.

rules for scoring 10 required AUs[3] having the following form (see [9]): "If the facial points representing the mouth corners move upwards, then score AU12".

3. Provide the required explanations. In brief, students should articulate what they have learned and reflect on the process and decisions that were entailed by the process. The relevant explanations concern the choice of the inference procedure and the manner in which optimal values for the thresholds used by the rules have been determined (usually in an automatic, iterative manner).

II. Neural Network vs. Case-Based Reasoning for Emotion Recognition

The second assignment is as follows: *Build two versions of an emotion recognition system, one using a neural network and one using case-based reasoning. The input to the system should be the AUs detected in the first assignment. The output should be one or more basic emotions: happiness, sadness, anger, fear, surprise, or neutral. Explain the advantages and disadvantages of each approach to emotion recognition.*

Participating teams of students usually tackled this problem in the following way.

1. Using the AU-coded videos of facial expressions of emotions from the Cohn-Kanade database [4], train a Backpropagation Neural Network for AU-based emotion recognition. For visualizing the training process use the available Visualization Tool (Figure 3). Given the training results choose a more appropriate number of the hidden layers and the related hidden neurons. Test the (best performing) trained Neural Network on previously unseen AU-coded videos from the MMI database [10].
2. Develop a case base as an incrementally self-organizing memory that allows event retrieval based upon the generalizations formed from prior input (as proposed in [11]). Represent each case (event) as a specific set of AUs. Group the events related to one specific emotion within the same dynamic memory chunk. Define the indexes associated with each dynamic memory chunk so that they comprise individual AUs and AU combinations that are most characteristic for the emotion category in question. To be able to classify a set of input AUs into an emotion category, develop a memory-chunk-index-based retrieval algorithm that will search the dynamic memory for similar cases, retrieve them, and interpret the input set of AUs using the interpretations suggested by the retrieved cases. Train the case base using the AU-coded videos of facial expressions of emotions from the Cohn-Kanade Facial Expression database (Figure 4). Test the trained case base on previously unseen AU-coded videos from the MMI Facial Expression Database.
3. Provide the required explanations. The main reason that students were asked to explain the (dis-)advantages of each of the utilized machine learning methods was to estimate whether they understood the difference between eager learning methods such as the neural networks and the lazy learning methods like case-based reasoning.

III. Person Identification and Remote Database Search

The third assignment is as follows: *Create a MAS that can perform person identification, emotion recognition, and search through a remote database to find the right person showing the right emotion. The input to the agent is an image of person A showing an emotion AE and an image of person B showing an emotion BE. The output should be an image from the MMI database of person A showing the emotion BE. Explain the choices made.*

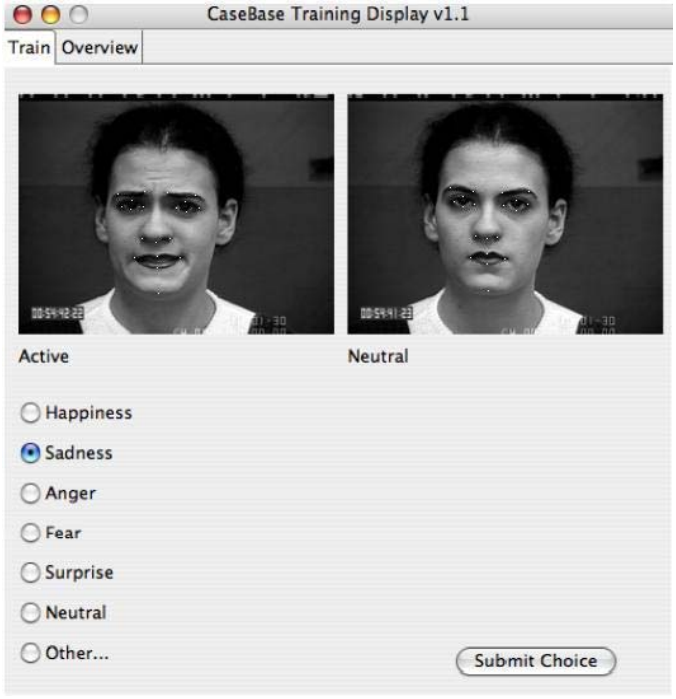

FIGURE 4
CASE BASE TRAINING INTERFACE AND ILLUSTRATION OF 20 CHARACTERISTIC FACIAL POINTS TRACKED IN EACH INPUT FACIAL VIDEO.

[3] There are in total 44 different AUs [2]. Each and every facial expression that the human face can display, can be described in terms of those 44 AUs. The 10 AUs to be recognized by the systems built by students are the ones that are typically displayed in facial expressions of emotions (e.g., frown, smile, etc.).

Participating teams of students usually tackled this problem in the following way.

1. Build an *Identification Agent* that identifies a person from an image of his or her face. Use the *Reader Agent* from the first assignment to trigger the facial point detector described in [9]. Use the characteristics of the rectangle formed by the inner corners of the eyes and the corners of the nostrils for person identification (as proposed in [15]).
2. Build a MAS for emotion recognition by reusing various agents developed for the first two assignments. Use the trained Neural Network instead of the trained Case Base since it requires less memory storage capacity.
3. Build the complete mobile MAS for person identification and emotion recognition and send it to the remote server where the MMI database resides.
4. Provide the required explanations. The most important one concerns the advantages of using the mobility concept – instead of downloading 20GB of image data, 200KB of MAS software is sent to the server where the MMI database resides, resulting in a tremendous reduction of Internet traffic.

CLASSROOM ASSESSMENT

The effectiveness of the proposed CBT program for teaching introductory Machine Learning was assessed by objective measures (test grades) and subjective measures (perceived satisfaction) of student learning. The latter was assessed by means of a standard questionnaire for eliciting students' attitudes toward computer science courses given at Delft University of Technology. The questionnaire solicits students' attitudes toward the course in general and their opinions about the related project (if any), the utilized tools and readings, the parts of the course that taught them most, and the ways the course could be enhanced. The questionnaire employs a 5-point Linkert scale ranging from extremely dissatisfied (1), via neutral (3), to extremely satisfied (5).

The CBT program explained in this paper has been introduced for the first time in 2005. Before that, the project part of the course concerned training various neural networks using different learning rules and explaining the observed behavior of the networks. In 2005, from 92 participants in the project, 78 filled out the questionnaire. The survey results are summarized in Table 1.

Table 1 shows that 92% of the students who filled out the questionnaire indicated that the utilized educational tool is affable and suitable for the goals of the course. As already concluded in our previous studies on using Fleeble to teach AI programming to novices [7], [8], the reason for such a score is that students perceive Fleeble as useful, easy to use, and affectively qualitative.

Table 1 shows further that the students perceived the CBT program as motivating and engaging. It also shows that they perceived the assignments useful for learning ML and AI programming. They also claimed that in the course of the presented project they enhanced their programming skills and acquired new, valuable, and applicable knowledge on the instructed subjects. They indicated that most of the learning occurred with the design and implementation of the multi-agent systems as required by the assignments. Overall, they were happy with the results of their work in terms of the performance and usefulness of the developed multi-agent systems. Table 1 also shows students' perceived usefulness of explaining what they have learned during assignments and reflecting on the process and decisions that were entailed by the process.

TABLE 1
STUDENTS' SATISFACTION AND PERCEIVED USEFULLNESS OF THE CBT PROGRAM. ALL QUESTIONS EMPLOYED 5-POINT LINKERT SCALE. THE PERCENTAGES IN THE TABLE SHOW THE PERCENTAGES OF FOURS AND FIVES.

Survey question	
The utilized tool (Fleeble) is suitable for the course	92%
The CBT program was motivating and engaging overall	92%
I gained new skills in ML and AI programming during the project	84%
I gained new knowledge in ML and AI during the project	86%
I found the 1st assignment useful for gaining new knowledge/skills	76%
I found the 2nd assignment useful for gaining new knowledge/skills	93%
I found the 3rd assignment useful for gaining new knowledge/skills	85%
I learned much during the analysis of the problems	33%
I learned much during the design of the solutions	81%
I learned much during the implementation of the solutions	95%
Reflecting on what I have learned during an assignment was useful	86%

As remarked by Schell [13], although students may frequently report a perception of learning more than in traditional courses, quantitative measures do not always confirm this perception. Hence, in order to assess objectively the usefulness of the employed CBT program, we performed intra-class objective measurements. The assessment measured the performance of students' answering the questions about rule-based reasoning, neural networks, case-based reasoning, and agent technology, and the performance of students' answering the other four questions (fuzzy logic, fuzzy expert systems, genetic algorithms, and foundations of AI and ML, which were not taught by means of the CBT program) on the final exam. The grade obtained for each question was compared with those obtained for the other questions by computing the ratio between the average score achieved by the students to the maximum score for each question. As can be seen from Figure 5, the case-based reasoning question rated the highest, the neural networks question was second, the rule-based reasoning question was third, the agent technology question was fourth, while genetic algorithms question, fuzzy expert systems question, and fuzzy logic question rated much lower. These results indicate that the specified programming assignments improved the students' ability to apply the acquired knowledge to novel problems. They also suggest that the proposed CBT program is highly suitable for the purposes of teaching basic AI concepts and ML techniques, including case-based reasoning, neural networks, rule-based reasoning, and the intelligent agents paradigm.

Finally, one may wonder why we did not conduct inter-class (between two classes) assessment concerning, for example, the classes of 2004 and 2005. The main reason is that we could not carry out this measurement without creating multiple simultaneous changes in the system. In other words, we could not match the two classes on various possible factors

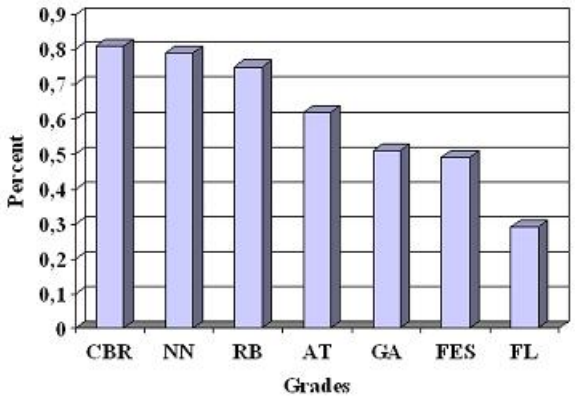

FIGURE 5
THE PERFORMANCE OF STUDENTS' ANSWERING DIFFERENT QUESTIONS ON THE FINAL EXAM. CBR: CASE-BASED REASONING. NN: NEURAL NETWORKS. RB: RULE-BASED REASONING. AT: AGENT TECHNOLOGY. GA: GENETIC ALGORITHMS. FES: FUZZY EXPERT SYSTEMS. FL: FUZZY LOGIC.

so that we could measure the educational effectiveness of one specific factor (e.g., the proposed CBT program). The two classes differed across many factors including the lecturer teaching the course, in-class lectures and activities, the content of the project, the utilized educational tools, and the final exam. Hence, although the fact that 73% of the participating students past the course in 2005 while only 49% past it in 2004 could be an indication of the educational effectiveness of the proposed CBT program, it could also be an indication of the educational effectiveness of any other newly introduced factor (lecturer, in-class activities, etc.).

CONCLUSIONS

The CBT program presented in this paper for teaching Machine Learning to second-year undergraduates represents a pragmatic implementation of active-learning philosophy. It blends the objectivist and the constructivist approach to introducing students to concepts of agent technology, expert-systems, neural networks, case-based reasoning, and mobility while epitomizing learning that is interactive, student-centered, exploratory, contextualized, intentional, reflective and collaborative. More specifically, teams of students (interactive, collaborative work) are presented with well-defined (objectivist approach), team-tailored assignments (student-centered) aimed at recognition of facial expressions and emotions from face videos and at retrieval of relevant information from a remote database (constructivist approach and contextualized, exploratory, intentional, reflective work).

The objective measures of student learning have indicated that the proposed computer-based active-learning method significantly increases the extent of learning compared with use of only the objectivist approach to teaching ML and AI programming to novices. In turn, the CBT program proposed in this paper can be widely useful for any other (introductory) ML course.

ACKNOWLEDGMENT

The authors would like to thank all CS and EE undergraduates who participated and evaluated the project in 2005. The work described in the paper was conducted while the authors were with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands.

REFERENCES

[1] http://www.acu.edu/cte/activelearning/whyuseal2.htm

[2] Ekman, P., & Friesen, W., Facial Action Coding System, Consulting Psychologist Press, 1978.

[3] Jonassen, D.H., & Rohrer-Murphy, L., "Activity theory as a framework for designing constructivist learning environments", *Educational Technology Research and Development*, Vol 47, No 1, 1999, pp. 61-79.

[4] Kanade, T., Cohn, J., & Tian, Y., "Comprehensive database for facial expression analysis", *Proc. Int'l Conf. Face and Gesture Recognition*, 2000, pp. 46-53.

[5] Lister, R., & Leaney, J, "Bad theory versus bad teachers: Toward a pragmatic synthesis of constructivism and objectivism", *Int'l Conf. of Higher Education Research and Development Society of Australasia Inc.*, 2003, pp. 429-436.

[6] McCracken, M., et al., "Multi-national, multi-institutional study of assessment of programming skills of first-year CS students*", SIGCSE Bulletin*, Vol 33, No 4, 2001, pp. 1-16.

[7] Pantic, M., Grootjans, R.J., & Zwitserloot, R., "Fleeble Agent Framework for teaching an introductory course in AI", *Proc. Int'l Conf. Cognition and Exploratory Learning in Digital Age*, 2004, pp. 525-530.

[8] Pantic, M., Grootjans, R.J., & Zwitserloot, R., "Teaching Ad-hoc Networks using a Simple Agent Framework", *Proc. IEEE Int'l Conf. Information Technology Based Higher Education and Training*, 2005, pp. 6-11.

[9] Pantic, M., & Patras, I., "Detecting facial actions and their temporal segments in nearly frontal-view face image sequences", *Proc. IEEE Int'l Conf. Systems, Man and Cybernetics*, 2005, pp. 3358-3363.

[10] Pantic, M., Valstar, M.F., Rademaker, R., & Maat, L, "Web-based database for facial expression analysis", *Proc. Int'l Conf. Multimedia and Expo*, 2005, pp. 317-321.

[11] Pantic, M., & Rothkrantz, L.J.M., "Case-based reasoning for user-profiled recognition of emotions from face images", *Proc. Int'l Conf. Multimedia and Expo*, 2004, pp. 391-394.

[12] Pantic, M., Zwitserloot, R., & Grootjans, R.J., "Teaching Introductory Artificial Intelligence Using a Simple Agent Framework", *IEEE Transactions on Education*, Vol 48, No 3, 2005, pp. 382-390.

[13] Schell, G.P., "Universities marginalize online courses", *Communications of the ACM*, Vol 47, No 7, 2004, pp. 53-56.

[14] Zhang, P., & Li, N., "The importance of affective quality", *Communications of the ACM*, Vol 48, No 9, 2005, pp. 105-108.

[15] Zhao, W., Chellappa, R., Phillips, P.J., & Rosenfeld, A., "Face Recognition: A Literature Survey", ACM Computing Surveys, Vol 35, No 4, 2003, pp. 399-458.

[16] Zwitserloot, R., & Pantic, M., "Agent Frameworks". In: M. Pagani (ed.), The Encyclopedia of Multimedia Technology and Networking, Vol 1, 2005, pp. 15-21. Hershy: Idea Group Reference.