Artificial Intelligence for Non-Majors at Multiple Levels

Andrea Pohoreckyj Danyluk
Department of Computer Science
Williams College
47 Lab Campus Drive
Williamstown, MA 01267
andrea@cs.williams.edu

Abstract

Over the past several years, many computer science departments have seen a decline in enrollments. This paper describes two courses – at the introductory and advanced levels – that hope to attract students to computer science through topics in Artificial Intelligence.

Over the past several years, many computer science departments have seen a decline in enrollments. As the program committee for this symposium has noted, AI topics have the potential to draw students back to computer science. This paper describes two ways in which we at Williams College are providing opportunities for students – in particular non-majors – to study AI topics. These are:

- An introductory course on AI and robotics for non-majors;
- An elective course on machine learning that is taught in a tutorial format.

While these courses are taught at very different levels, they share the following:

- Both have the potential to draw non-majors into more than one computer science course.
- Both can draw students into computer science courses fairly early in their college careers.
- Both have ties to the Cognitive Science program, which provides an extra means of advertising the courses to students who might not consider computer science.

The introductory course for non-majors has no prerequisites and can therefore be taken by students beginning in their first semester. As will be discussed below, students in the course often go on to other computer science courses. The advanced course has prerequisites, so non-majors interested in it must also enroll in other computer science courses. The prerequisites are fairly minimal, however, so the non-majors aiming to take the machine learning course need not begin planning for it immediately. This makes it realistic to believe that they will have time to benefit from faculty and peer advising that would point them to the course. Both of these courses are electives for a concentration in Cognitive Science, and since this puts computer science "on the radar" for concentrators, this encourages enrollment in computer science courses more generally. These points will be revisited in sections that discuss the individual courses.

Links to course materials for both courses are provided below. As the introductory course for non-majors is described in detail elsewhere (Danyluk 2004), this paper provides more details on the machine learning tutorial course. In the section on the introductory course, the details given primarily highlight differences from the previously published version of the course.

Introductory AI for Non-majors

The traditional introduction to our discipline is a programming course. In recent years, however, more and more computer science faculty have become dissatisfied with introductory courses that teach programming but do not actually introduce students to the important ideas of computer science. At Williams, for example, we have begun to experiment with a CS1 course that simultaneously teaches students Java programming as well as important concepts such as abstraction and representation through lectures and exercises on computer networks. (Murtagh 2007). Many others have begun to experiment with introductory courses that are introductions not just to programming, but to computer science. Faculty at Middlebury College, for example, have devised a course that brings together their programming introduction and a breadth-first introduction to the field. (Briggs 2007)

Even when it is desirable to provide an introductory programming course (say, in the case where CS1 is a service course for other departments), it is still possible to devise other rigorous, interesting courses for majors and non-majors. (Koffman *et al.* 2007) Here I describe briefly one such course that uses AI and robotics as core themes.

The goals that inspired the design of the course were as follows:

- It should introduce students to fundamental questions and issues of computer science;
- It should make all students feel comfortable in the course, regardless of their background in science;
- It should give students enough practice with programming and problem solving that they can determine whether they have the aptitude to go on in computer science;
- It should motivate some students to consider becoming Computer Science majors.

Many of the ideas for this course were taken from a similar course offered at Swarthmore (Meeden ; Kumar & Meeden 1998).

Course Format

In this section I describe the format of the course, and in the following two I give more details on the lecture and lab syllabi. Complete course information, including the syllabus, lecture notes, and lab write-ups can be found at http://www.cs.williams.edu/~andrea/cs108.

The AI course for non-majors is taught over twelve weeks. It meets for three fifty-minute lecture sessions each week. There is also a weekly lab session in which students construct and program simple robots. The lab officially meets for ninety minutes, but students are welcome to stay longer. Indeed, in most weeks they can expect to come back to lab to finish the weekly assignment.

In addition to lab exercises, students do readings for each class. The readings include textbook readings, position papers on various aspects of AI, philosophy, and psychology, as well as fiction. During discussion weeks, students submit a brief response to the readings for the week. Students also complete four problem sets.

Students can work in small groups on the lab exercises, though their lab reports, reading responses, and problem sets are to be done independently.

Lecture Syllabus

Roughly half the semester is spent on topics specifically related to robotics and the programming that students will be doing in lab. The second half of the semester is devoted to more general topics within AI.

Topics covered in the first half of the semester include:

- applications of robotics,
- challenges for robotics (such as unpredictability of actions in the real world),
- robot sensors and effectors,
- classical architectures and STRIPS-style planning,
- behavioral approaches,
- navigation and motion planning,
- configuration spaces.

Five lectures are devoted to programming in Interactive C. This material is presented in a just-in-time fashion – students learn just what they need to complete the current lab exercise.

Topics covered in the second half of the course include units on knowledge representation and reasoning, search, game playing, machine learning, natural language processing, and computer vision. We also discuss topics such as robot ethics, creativity, and the nature of intelligence.

Lab Syllabus

As indicated above, in the lab students build and program simple robots. The platform used is the Handy Board (Martin) robot controller. Students begin by soldering and wiring the motors and sensors they will use for the rest of the semester. In their second lab, they test the wiring they completed the week before. By the third week, they have built the basic robot. And then the programming fun begins. Each week students program simple behaviors: bumper sensing to navigate around obstacles, sonar to avoid obstacles altogether, line following, maze navigation comparing the behaviors programmed earlier in the semester, and, finally, a multiweek project that involves building and programing a trashbot, which seeks out trash (in the form of 2-litre soda bottles) and uses a gripper to haul them away.

Benefits of the Course

Rather than discussing the general pedagogical benefits of the course, I will focus here on the benefits of the course at attracting students to computer science. Of the thematically based non-major courses we teach at Williams, this one draws the fewest students. Since the course regularly receives strong evaluations, we believe that this is due to a combination of two factors.[1] First, the course description for this non-major course most clearly states that programming will be an important element of the course. Second, this course is offered less regularly than our other non-major courses, and so it has less opportunity to develop a following.[2] The last offering in Fall 2006 had only four students enrolled. However, of the four, three were women and two of the students were African-American. Of those four, one student has gone on to become a Computer Science major (she was originally planning to major in Mathematics) and two of the others have taken more computer science (one is a senior English major and the other is interested in Cognitive Science and Art History.)

Table 1 give the enrollment history of the course since 1999. (The course was offered twice before that, but without the robotics component.) In the table, the column labeled "Students" gives the number of students enrolled in the course, though both the Fall 2000 and Spring 1999 offerings have been adjusted to reflect students who subsequently withdrew from the college and therefore could not take more computer science. The next column gives the number of students who decided to major in computer science after taking the course. The next gives the number of other students who took computer science in addition to this one course. The next column gives the number of regular computer science courses taken by these non-majors. It is important to note that this number reflects courses taken both before and after enrollment in the non-major AI course. The final column gives the number of Winter Study enrollments in computer science. The Winter Study Period is a 3.5-week January term in which each Williams student is required to take one course (which might be anything from auto mechanics to critical analysis of Shakespeare's *The Tempest*).

It is important to put the numbers in the table in context. Williams is a liberal arts college of approximately 2000 stu-

[1] It would be interesting and valuable to survey students in our other non-major courses to get a sense of why they do not elect to take the AI-themed course.

[2] Other teaching obligations make it impossible for us to offer the course as regularly as we might like.

When	Students	Majors	Other CS	Regular Courses	WSP
F06	4	1	2	3	0
F04	10	0	8	9	4
F00	15	2	6	12	0
S99	14	0	5	10	0

Table 1: Enrollments in non-major AI course and beyond.

dents. Because of the liberal arts emphasis, students typically do not come to the college planning to major in computer science. We have a healthy number of majors – on average 15 in each class – but this is not as high as one would expect at a larger school or at one with an engineering emphasis. While the numbers in Table 1 are not extremely high, they add a non-trivial number of enrollments to our department.

As mentioned above, the final column of the table gives the number of enrollments in computer science Winter Study courses. In Fall 2004, three of the students in the non-major AI course asked whether our department might offer a Winter Study course on robotics, which would allow them to continue the work they started that semester. Thus three of the four enrollments given are those. The Winter Study course that grew out of the non-major AI course has now been offered twice, reaching the enrollment limit each time (15 in the first offering, 16 in the second). Of the 31 students enrolled, 20 were non-majors.

A Machine Learning Tutorial Course

This section describes an elective in computer science on the topic of machine learning. I start by describing the audience for the course as well as the unusual course format that makes it possible – indeed, relatively easy – to tailor the course to students with a wide range of backgrounds and interests.

Audience and Format

This course was designed primarily for majors, but the first offering in Fall 2005 attracted students from Economics and Mathematics as well. Their interest in the course was inspired by their desire to understand the implementation of algorithms behind the data mining techniques they were studying in courses in their respective majors. As an upper-level elective, the course clearly has computer science prerequisites, but these are minimal. They include Data Structures (a standard CS2 course) and Discrete Mathematics (though this requirement can be waived with permission of the instructor).

Teaching a course with an audience of varied backgrounds presents interesting challenges. The format of this course on machine learning, however, lends itself well to a group with varied interests and abilities. As described in the college course catalog, the tutorial course format offers Williams students "a distinctive opportunity to take a heightened responsibility for their own intellectual development." In addition, tutorials provide for the instructor a natural mechanism to tailor the material for small groups of students.

Typically, tutorial courses have an enrollment limit of ten. Students meet with the instructor in pairs; each pair meets with the instructor once a week for approximately one to one and a half hours. (Thus if a course is full, the instructor meets with five pairs of students each week.) Weekly sessions are conducted differently in various disciplines. In computer science, we typically assign readings for the week as well as some combination of other work that might include problem solving, proofs, implementation, critical analysis of data, or paper critique. In the tutorial sessions students take turns presenting the material, including their solutions to the weekly assignment. The student not presenting is expected to provide critique, which might include anything from catching errors to explaining different ways of thinking about the problem to feedback on presentation clarity. The instructor sets the agenda for the course, but the students take primary responsibility for learning the material. In the process, they develop critical thinking and presentation skills, among others.

Students at Williams are not required to take tutorials, but many do. In 2006-2007 over 60 tutorials were offered at the College. In Computer Science we attempt to offer two tutorials each year. In the class of 2006, only 2 of 9 Computer Science majors graduated without taking a tutorial for the major. In 2007, only 3 of 16 majors graduated without a tutorial for the Computer Science major. Of those who took tutorials, some had three, four, and even five tutorials for the major. We anticipate that by the time they graduate, only 3 of 12 majors in the class of 2008 will graduate without taking a tutorial. While tutorial courses would not appeal to all students at all institutions, there are clearly many students are are interested in the opportunity to learn in this independent way. In our experience, it is not just the strongest students who elect to take tutorials. Students at all levels find them an interesting way to learn new material. They enjoy these courses for the contrasting combination of independence and (almost) one-on-one "classroom" attention of the weekly meetings.

While independence and individuality contribute to the positive experience that students have with tutorials, the fact that material can be tailored to their interests and background is also extremely valuable. The following two sections describe the course content in more detail, but this is followed by a discussion of ways in which this mode of teaching and learning might be adapted for other institutions.

Course Syllabus

This twelve-week course covers machine learning with a distinct emphasis on classifier-learning algorithms. The weekly topics are as follows:

- Introduction to Machine Learning; Linear Models and Perceptrons
- Probabilistic Models: Naive Bayes
- Regression
- Decision Trees
- K-Nearest Neighbor (A short topic due to midterm reading period)
- Neural Networks
- Support Vector Machines
- Evaluation Methodology
- Learning Theory
- Bias/Variance and Ensemble Methods
- Clustering
- Applications

Below I describe the assignments for three selected weeks to illustrate the range of assignments and to have a basis for discussing the ways in which meetings with student pairs can be tailored to their understanding of the material. Complete details, including assignment handouts and readings are available at http://www.cs.williams.edu/~andrea/cs374.

Sample Assignments

Introduction, Linear Models, Perceptrons The introductory assignment for the course is somewhat ambitious, as it is important to introduce students to machine learning in general, give them a flavor of at least one specific method, and also give them an early sense of the types of assignments they will be asked to complete. The assignment begins by having students read the first two chapters in their primary text (Alpaydin 2004). The students then do several book exercises to reinforce their understanding of the reading. And then they are ready to look at their first simple model. The reading for this topic, also taken from the text, describes the geometry of the linear discriminant as well as the perceptron learning algorithm, one simple approach to learning a linear discriminant.

The section on the geometry of the linear discriminant includes several places where major steps in a derivation are skipped or where claims are made without detailed explanation. It is important that students feel empowered to understand the large leaps made in upper level textbooks, so some of the exercises have students fill in the missing steps and explanations. Having completed these exercises, students then derive the more general update rule used in backpropagation.

Because perceptrons are such simple models, the week's assignment ends with an application paper that makes use of them, to illustrate that even simple models can be valuable. (Fawcett & Provost 1996)

All students must make an attempt to do all parts of the assignment, but the meetings with the instructor can be tailored for each pair of students. The weakest students might only get through their solutions to the first exercises and half of the linear discriminant exercises. This is fine, as they will return to neural networks later in the semester. Stronger students will complete presentations of their solutions to all of the problems. Still stronger students will have time to discuss the details of the paper, rather than simply acknowledge its interestingness as an illustration of the utility of a simple approach.

Decision Trees This week the focus is on a specific decision tree learning algorithm, C4.5 To get the students started, the assignment suggests that they first skim three sections in Alpaydin's Machine Learning text. Then it recommends Chapter 3 in Mitchell's text (Mitchell 1997) as their primary source for the topic. It suggests three other sources, just in case the students feel they need additional readings. These include sections in Russell and Norvig's AI text (Russell & Norvig 2003) as well as two readings by Quinlan (Quinlan 1986; 1993). Since the students will be doing no implementation during this week, the assignment suggests that they run the decision tree implementation in WEKA (Witten & Frank 2005) in order to get a feel for how decision tree learning algorithms work.

The first few exercises are quite straightforward. They have students draw the decision boundaries for a simple tree that splits on real-valued attributes, and also simulate the construction of a decision tree on a small data set. The exercises then become slightly more interesting. Students are asked to consider trees in which mixed sets of examples remain at the leaf nodes and show that, for example, the labeling that is based on majority classification minimizes absolute error. They then show that the entropy function is concave in order to demonstrate that every attribute has nonnegative information gain.

Once the students have a reasonably solid grasp of decision trees and C4.5, they read a research paper on skewing, which addresses a specific problem that can arise in decision tree learning. (Page & Soumya 2003)

In the actual tutorial session, the weaker groups spend most of their time presenting their solutions to the problems, while the stronger groups are able to quickly bypass the exercises in order to spend more time discussing the research article. In fact, students were sufficiently interested in knowing *why* skewing works (the paper shows empirically that it works, but does not formally demonstrate why) that I was able to modify their subsequent learning theory assignment to include a follow-up paper. (Rosell *et al.* 2005).

Evaluation of Learning Algorithms This week begins, as usual, with a reading to introduce students to background material, which, for this topic, includes some elementary statistics. There are two readings given, and the assignment suggests to the students that one of them has more clear explanations, but that the other covers more material. The assignment then asks that students complete several book exercises to test their understanding of the reading. Next the assignment takes the students through the process of setting up a machine learning experiment. They evaluate five algorithms (all of which they have already studied) and apply them to at least four data sets, which span a range of sizes

and complexity. The product of their work is an evaluation report.

Once they have completed the exercise, which uses accuracy on the test set as the measure of goodness, the assignment asks students to read a paper that challenges the use of accuracy estimation as the measure of quality of a learning algorithm (Provost, Kohavi, & Fawcett 1998).

Adapting to Other Settings

Teaching a course as an upper-level elective that will address the interests and experience of a diverse audience can be a real challenge. The course described here is taught in the tutorial format so that the weekly "classroom" experience can be tailored to very small subsets of students. Since the tutorial format is not a standard teaching format at most institutions (and would be entirely impractical at a large school with high enrollments), there are a couple of different ways to adapt the course.

With relatively low enrollments, the course described above can be taught as a guided independent study course. A professor might meet with students in pairs, as in the tutorial model, or in slightly larger groups. This might require more contact hours than a standard lecture course, but the tradeoff of having well-crafted exercises that guide students through the material makes up for this – it simply results in students needing less help in between lectures.

At a large school with significantly higher enrollments, students might be assigned to conference sections based upon their background. The focus of the sessions, then, could be tailored as in the small tutorial meetings.

In either case, the course would be attractive to a wide range of students, and still be manageable for the instructor.

Benefits of the Course

Machine learning is increasingly becoming interesting to statisticians, economists, political scientists, biologists – indeed, anyone who hopes to perform analyses of large data sets. As a result, there is great potential to bring students into computer science by providing them with an opportunity to learn about new tools and algorithms or simply to learn more about the algorithms they have already been using as parts of established toolkits in their own disciplines. It is important that these students have some computer science background, but in order to lower the barriers to taking the course, prerequisites must be minimal. As noted above, the prerequisites for the machine learning tutorial are Data Structures and Discrete Math, though it would be reasonable to remove the discrete math requirement. Thus the hurdles that students need to jump before entering the course are, while not trivial, not insurmountable. Students must begin to take these courses fairly early, which is beneficial as they might decide to become majors along the way. At the same time, because the prerequisite chain is not long, they do not have to begin immediately upon entering college. This gives them time to establish their interests and get to know both faculty and peers who can advise them to follow this path. Indeed, this is the approach we have followed with our elective on AI. In the five offerings of AI since Spring 1996, the course has had an average enrollment of 21.4, with an average of 2.4 non-majors taking the course each time (3.4 if we take into account upper class students with declared majors who later changed to or added computer science).

As indicated in the introduction, both the machine learning tutorial and the non-major AI course are listed as electives for the Cognitive Science concentration. (The advanced AI course is as well.) This very explicit visibility encourages students to consider computer science even in the absence of a formal advising program. Since the Cognitive Science program began in Fall 2002, 6 concentrators have graduated. Of these 6, 2 were computer science majors. Of the remaining 4, 2 took computer science courses. Eight more concentrators will graduate by 2009. Of these, 3 are computer science majors. Of the remaining 5, 2 have already taken computer science courses.

Summary and Additional Thoughts

Artificial Intelligence is a topic that fascinates many. Furthermore, the AI research community has made significant strides, so that AI technology can genuinely be applied to a variety of other fields. Given this, it is natural to imagine that students can be drawn into computer science through their interest in AI topics. This paper has presented two courses – one at the introductory level and the other at the upper level – that have appeal to an audience beyond the computer science major.

A significant (though not sole) contributor to the success of these courses (as measured by the students they draw into computer science) is the extent to which faculty in other disciplines recognize the courses as worthwhile places for their students. Forging relationships with faculty in areas such as Cognitive Science, Economics, and Statistics, among others, is an important component of the process of raising our enrollments.

References

Alpaydin, E. 2004. *Introduction to Machine Learning*. Cambridge, MA: MIT Press.

Briggs, A. 2007. Personal Communication.

Danyluk, A. 2004. Using robotics to motivate learning in an ai course for non-majors. In Greenwald, L.; Dodds, Z.; Howard, A.; Tejada, S.; and Weinberg, J., eds., *Accessible Hands-on Artificial Intelligence and Robotics Education: Papers from the 2004 AAAI Symposium*. AAAI.

Fawcett, T., and Provost, F. 1996. Combining data mining and machine learning for effective user profiling. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*.

Koffman, E.; Ellis, H.; Kelemen, C.; White, C.; and Wolfman, S. 2007. New paradigms for introductory computing courses. In *SIGCSE 2007, Proceedings of the Thirty-Eighth SIGCSE Technical Symposium on Computer Science Education*.

Kumar, D., and Meeden, L. 1998. A robot laboratory for teaching artificial intelligence. In *Proceedings of the*

Twenty-ninth SIGCSE Technical Symposium on Computer Science Education (SIGCSE-98).

Martin, F. The handy board. World Wide Web, URL is http://lcs.www.media.mit.edu/groups/el/Projects/handy-board/.

Meeden, L. World Wide Web, URL is http://www.cs.swarthmore.edu/meeden/.

Mitchell, T. 1997. *Machine Learning*. Boston, MA: McGraw-Hill.

Murtagh, T. P. 2007. Weaving cs into cs1: A doubly depth-first approach. In *SIGCSE 2007, Proceedings of the Thirty-Eighth SIGCSE Technical Symposium on Computer Science Education*.

Page, D., and Soumya, R. 2003. Skewing: An efficient alternative to lookahead for decision tree induction. In *Proceedings of IJCAI-03*.

Provost, F.; Kohavi, R.; and Fawcett, T. 1998. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of ICML-98*.

Quinlan, R. 1986. Induction of decision trees. *Machine Learning* 1(1).

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

Rosell, B.; Hellerstein, L.; Ray, S.; and Page, D. 2005. Why skewing works: Learning difficult boolean functions with greedy tree learners. In *Proceedings of ICML-05*.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. San Francisco: Morgan Kaufmann.