Mission to Mars - USD 383 Summer STEM - Day 4

Summer 2017

Learning Objectives

- Students will discuss artificial intelligence and how to determine if a computer is intelligent.
- Students will experience how a neural network is built using training data.
- Students will build a working video game AI based on PacMan.
- Students will use their knowledge of AI to write a program to avoid random obstacles.

Resources

- Slides: http://people.cs.ksu.edu/~russfeld/presentations/stem2017/day4.html
- Scratch Website: http://scratch.mit.edu
- Information on Turing Test: http://csunplugged.org/the-turing-test/
- Pacman AI Behavior: http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior
- Pacman on Scratch: https://scratch.mit.edu/projects/65404312/
- Pacman Solution on Scratch: https://scratch.mit.edu/projects/90692358/
- Mars Pathfinding on Scratch: https://scratch.mit.edu/projects/113105354/

Lesson Setup Before Class

- Log on to computers using STEM accounts
- Print Cat & Dog handouts
- Make sure **PacMan Starter** is available on Scratch website.
- Make sure **Mars Pathfinding** is available on Scratch website.

Schedule

- 9:00 - Welcome & Icebreaker
- 9:15 - Intro to AI & Decision Making
- 9:30 - Cat or Dog activity
- 9:45 - PacMan AI
- 10:10 - Break
- 10:15 - Mars Rover Pathfinding
- 10:45 -STEM Survey
- 11:00 - Wrap-Up Discussion

Lecture Notes

1. [**Icebreaker**] Take a minute to review what was learned yesterday. Some good questions:
    a. Convert some numbers back and forth between binary
    b. What other numbering system besides binary did we learn? How does it work?

    c. What could happen if a message gets jumbled up when we send it? How can we fix that?

2. [**Video**] Today we are going to talk all about artificial intelligence. Before we begin, here's a short video from PBS introducing the subject with some of the world's renowned experts in the field. (Watch through 2:05, first section)

3. [**Video 2**] Here's another video about a robot that is programmed to learn just like a human does. Do you think that would be a good way to build an artificial intelligence?

4. [**What is Artificial Intelligence**] So, what is artificial intelligence?
    a. Discuss. A good definition is "the study and design of intelligent agents."

5. [**Intelligent Behavior Diagram**] One problem with AI is that it forces the computer to mimic all human behavior, not just intelligence. Things such as slow response times, typos, and commonly held misconceptions that aren't true are all examples of human behavior that might not be considered intelligent, but it is still something a successful AI agent must possess to pass the test. Likewise, it must also act as if it cannot solve some problems that are perfectly within its abilities, simply because those problems are unsolvable by a human's intelligence.

6. [**Alan Turing**] That was the problem that Alan Turing ran into. He was deeply interested in the field, but unfortunately there was no good way to determine if a system was truly "intelligent" at the time. In 1950, he wrote a paper called "Computing Machinery and Intelligence" which he opened with the following statement: "I propose to consider the question, 'Can machines think?'" In that paper, he describes one way to test a machine's intelligence, which is now known as a Turing Test

7. [**Turing Test**] The basic idea of a turing test is as follows: you have a person in a room with a computer capable of text-based chat. In another room is either a computer or a human that responds via the text-based chat system. The computer will try to pass itself off as a human being, by responding to the prompts from the tester. The tester then must determine if he or she was conversing with a computer or a real person. So far in history, no computer has completely passed the Turing test, but many have come very close at times.
    a. So, now that we've seen the turing test, can anyone think of some problems it may have?

8. [**Chinese Room**] Another problem with the Turing Test is highlighted in the Chinese Room thought experiment, as proposed by John Searle in 1980 in his paper "Minds, Brains, and Programs." In this setup, an English speaking person is placed in a room with sufficient supplies and a set of instructions completely written in English that directs him to accept Chinese language characters as input, and output a response of Chinese

characters. On the other side of the wall is a native Chinese speaker performing a "Turing Test," and in this case that person is convinced that the person on the other side of the wall is indeed a human. However, the computer, being a human that only speaks English, is blissfully unaware of the conversation taking place in Chinese. So, is the machine "intelligent" or merely so advanced at following instructions as to appear "intelligent?"

9. [**Strong AI vs. Weak AI**] This leads to the endless debate between Strong AI and Weak AI. When you think of AI in movies, this is usually "Strong AI" which is designed to completely mimic or surpass human intelligence. Unfortunately, at this time Strong AI is not a reality, and some debate that it is even possible. Most of the AI that we deal with today is a form of Weak AI, also called Narrow AI, which is designed to perform only a subset of intelligent actions.

10. [**Marvin Minsky**] To dive a bit deeper into the topic of AI, we're going to look at a very unique tool called Neural Networks. In 1969, Marvin Minsky, one of the founders of MIT's AI lab, wrote a book called *Perceptrons* that laid the groundwork for the idea of neural networks.

11. [**Artificial Neural Networks**] The idea behind neural networks lies in the power of individual "neurons" and the connections between them. Each neuron is capable of doing a certain task, and then its output is passed on to other neurons. The strength comes in the form of the connections between the neurons. If one tends to give correct answers to a problem, other neurons will be more likely to use its output based on the strength of the connection between them. The process of strengthening good connections and weakening bad ones is how a neural network is able to "learn" how to do things.

12. [**Neural Network Activity**]
    a. Each student gets one of the half-sheet handouts. They are directed to only look at their own sheet and not share.
    b. Around the room, post numbers 0 - 10
    c. For each picture, ask the class to vote whether the overall picture is of a Cat or a Dog. I usually have them close their eyes to prevent cheating. Also, remind them that it is OK to get this wrong (some students are very self-conscious about being wrong at this age).
    d. Use the Excel document to show the full picture.
    e. At the end, any students who got it right move up to the number that represents how many they've gotten correct so far. (So, if they have 3 correct, they stand by number 3). The students at the higher numbers get "more votes" or higher weight than the lower students.
    f. At the end, let the students vote one last time with their eyes open. Hopefully they should see that most of the students who get it right are at the higher

numbers. You can also use the Excel sheet to record the number each student is at based on the X-Y coordinates on the handout. Hopefully the higher numbers should be centrally located and the lower numbers are around the outside.

g. See **StudentTeach.mp4** for a short example of how this works.

13. [**Camouflaging Tanks**] Refer to story here: http://neil.fraser.name/writing/tank/
    a. Trained a neural network with pictures of tanks hiding in trees and pictures of just trees
    b. It worked well for all the original photos, but when they brought in a new set of photos, it was totally random
    c. The reason: the original photos had all the tanks taken on sunny days, and all the trees taken on cloudy days. They had built a machine to determine if it was sunny or not!

14. [**MarI/O**] Here's another use of Neural Networks - to play a video game. This video does a great job of explaining how neural networks can be evolved to complete any task.

15. [**Pacman AI**] Now that we know a little bit about AI, let's see if we can build one in Scratch. One of the simplest and yet most interesting AIs can be found in the old game Pacman. How many of you have played Pacman? There are 4 ghosts - Inky, Blinky, Pinky, and Clyde. Let's take a look at how to write their AI:
    a. Direct the students to load the **Pacman** starter scratch project: https://scratch.mit.edu/projects/65404312
    b. The students should click the "See Inside" button to see the code. If they have a Scratch account, they can also click the "Remix" button to save it to their account.
    c. The AIs in this project use a simple perceptron model. They first perceive their surroundings, then act based on that perception. To complete this project, follow these steps:
        i. **Blinky:** Blinky always moves toward Pacman. So, in the perceive stage, it must determine which direction to move. To do this, I usually draw a circle on the board and divide it into 4 quadrants with an X. Then, each line must be labeled, with 0 being up. So, the lines will be 45, 135, -135 and -45, going clockwise from 0. Label the quadrants up, right, left and down. In the code, we must point Blinky at the direction of Pacman, then analyze our direction variable to determine what quadrant to move to. It is easiest to go from least to greatest. So, you'll have something like:

```
If direction > -135 and direction < -45
    set Direction to Pacman to left
else
    If direction > -45 and direction < 45
        set Direction to Pacman to up
…
```

Once you have the direction set, the Act phase is simply to move in that direction. The Move block accepts directions in lower case (left, right, up, down). See **Blinky.png** in the Google Drive for an example.

ii. **Pinky**: Pinky always moves to a point a few spaces in front of Pacman, which is represented by a sprite called “PinkyPoint.” It uses much the same algorithm as Blinky. In fact, it is simple to just drag the blocks attached to “perceive” in Blinky and drop them on Pinky, then attach them to the perceive block. (DO NOT drag the Purple “Define perceive” block along with it; that will break things). Then, just change the set blocks to use the “Direction to Point” variable instead, and have Pinky move toward that point. See **Pinky.png** in the Google Drive for an example.

iii. **Clyde**: Clyde is a bit different. He will move toward Pacman most of the time, but if he gets too close he will retreat toward a point in the lower left corner of the screen, represented by a sprite called “ClydePoint.” So, his code is a combination of both Blinky and Pinky. So, just drag and drop the two perceive sections (again, DO NOT include the purple “Define perceive” blocks). In the act section, just include an If block to check if Clyde is within a set distance of PacMan (I’ve used 100). If it is too close, move toward ClydePoint, else move toward PacMan. See **Clyde.png** for an example.

iv. **Inky:** The AI for Inky is much more difficult to code in Scratch. But, if students want to experiment, all they have to do is attach blocks below the “When green flag clicked” together to active it, then they can code their own Perceive and Act phases.

d. See the **Pac Man Solution** file on Scratch for a complete example: https://scratch.mit.edu/projects/90692358

16. [**Mars Rover Pathfinder**] For the last project, we need to get Mark Watney back home. To do that, he has to traverse the Martian terrain, but there are obstacles in his way. We are going to write a program to use AI to help him find a way around the obstacles.

a. Direct the students to load the **Mars Pathfinding** starter scratch project: https://scratch.mit.edu/projects/113105354

b. The students should click the “See Inside” button to see the code. If they have a Scratch account, they can also click the “Remix” button to save it to their account.

c. For this project, there are 3 randomly generated circular obstacles between the rover (lower left) and the goal (upper right). Students can press the Space key to move the obstacles, but once they start they should not change the obstacles for a while. The rover must make it to the goal in as few steps as possible. Here are the basic steps to complete the project

i. First, have the students simply point towards the goal and put the custom Move block into a Forever block, and see what happens. By default, it will move toward the goal, but it will stop once it hits a rock.

ii. Once it hits an obstacle, we must figure out which one using some If blocks. See **Pathfinder1.png** for an example. At the same time we only want to move if we aren't touching an obstacle, so it needs to go in the last Else block.

    iii. In the case that we've hit an obstacle, we should find the direction to the center of the obstacle, then see which side would be shorter to go around. We can do that by comparing that direction with 45 degrees. If it is less than 45 degrees, we should turn right, else we should turn left. We can decide how many degrees to turn and how many steps to make in that direction before turning back toward the goal. I usually turn 90 degrees and move 100 steps before turning back. Students can adjust those numbers to make their program more efficient. See **Pathfinder2.png** for an example.
    iv. That idea can be generalized for all 3 obstacles.
    v. Finally, the students should add once more If block to detect if the goal has been reached and stop the program at that point.
    vi. Once that is done, encourage students to modify their solutions by adjusting the angles and movements to reduce the number of moves (reported in the upper left corner).

  d. See **MarsPathfinder.mp4** for an example of how to complete this program.

<<<**There are some videos at the end of the slides for time filler if needed**>>>

17. [AI Today] Some examples of AI today:
18. [Deep Blue] Deep Blue was a computer created by IBM that plays chess. In 1996, over 20 years ago, it beat the world's reigning chess champion, Garry Kasparaov. It was once of the greatest milestones in AI, and today computers are becoming better and better at playing even more difficult games such as poker and go.
19. [Watson on Jeopardy] In 2008, another IBM computer called Watson participated in several rounds of the Jeopardy! game show against two of the greatest champions of all time. Watson won quite heavily, but not without making a few mistakes such as the one shown here in the video.
20. [2 AI Chatbots Talking] Of course, AI still have a long ways to go before it can interact just like a human would. This video shows one of the most advanced AI chatbots, Cleverbot, talking to itself. As you can see, the conversation really doesn't get very far.

21. [**Reflections**]
    a. What did we learn about computer programming this week?
    b. What was the most interesting thing we did?
    c. Do you think you could do more with computers? What other things would you like to learn?