Julie Townsend
CS229 12/15/06

Machine Learning with a Lego Mindstorms Robot

1.0 Summary

In this project, a machine learning algorithm was implemented on a Lego Mindstorms robot. Via supervised learning, a robot was trained to follow a path represented by a black line on a white background. The online perceptron algorithm was implemented such that training inputs supplied by the operator triggered algorithm updates. Testing proved that the robot could be successfully trained to follow either the right or left edge of the black path. Successful results were repeatable, and in all cases the robot was able to complete the line-following task successfully with an average of 22.3 training errors. The robot and path used for training are shown in Figure 1.

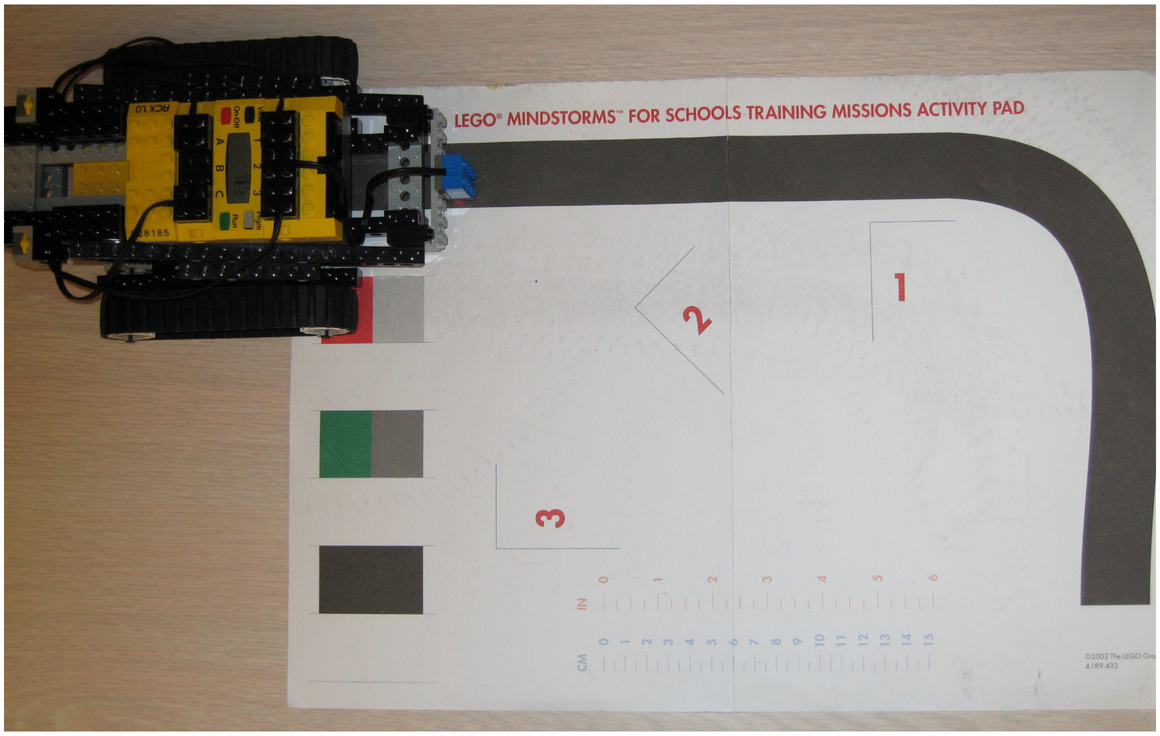
LEGO® MINDSTORMS™ FOR SCHOOLS TRAINING MISSIONS ACTIVITY PAD
1
2
3
IN
CM

Figure 1: Activity Pad used for training tests

2.0 Background

Lego Mindstorms for Schools is a kit available to teachers intended to introduce grade school students to robots and programming. The kit provides the parts necessary to build a variety of simple robots, like wheels, tank treads, motors, gears, an embedded computer, and a few sensors. It also provides Robolab, a graphical interface to create programs for the robots.

PUMAS (Practical Uses of Math And Science, http://pumas.jpl.nasa.gov/) is a collection of one-page examples of how math and science topics taught in K-12 classes can be used in interesting settings, including everyday life. The examples are written primarily by scientists and engineers, and are available to teachers, students, and other interested parties via the PUMAS Web Site.

Through my workplace, I have volunteered to create lessons for PUMAS that use the Lego Mindstorms for Schools kit. From this class project, I intend to create a lesson for grade school students in machine learning.

The selection of the Lego Mindstorms for Schools robot as a platform for this project imposed limitations on the scope of the learning task. The sensor set provided with the kit is rather limited, consisting of two light sensors and two touch buttons. The computer has only three output ports for motor control and only three input ports for incoming sensor data. The programming interface is also very limited. A palette of icons represents all the available functions and modifiers implemented in the Robolab interface. Loops and conditionals are available, but math capabilities are very limited. Only addition, subtraction, multiplication, division, and a few logical operators are supported, and all are performed with integers only.

3.0 Task selection

The starting point for task selection was the set of lesson plans provided with the Lego Mindstorms for Schools kit. Choosing from this set ensured that the robot would be capable of performing the selected task. One of the sample tasks provided with the kit was a simple algorithm for line following. This algorithm implementation is shown in Figure 2.

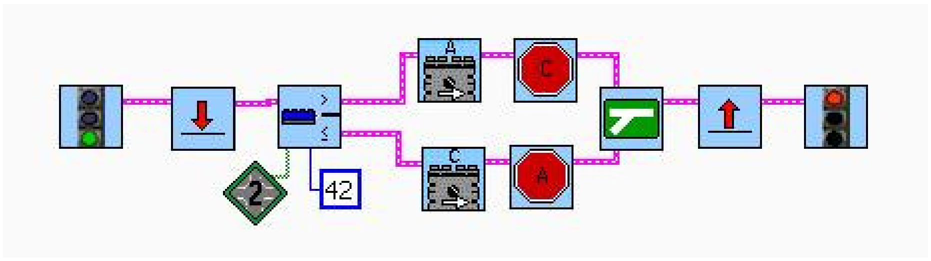

Figure 2: Robolab Line Following Program

The task consists of following the right edge of a black line based on data from a light sensor. In the Robolab implementation, the robot reads the light sensor on input port 2 and compares the value to a hard-coded threshold of 42. If the value is above the threshold, the sensor is over black. The program turns the left tank tread on and the right tank tread off, turning the robot right, toward the edge of the line. Similarly, if the value is below the threshold, the robot turns left. This cycle repeats until the program is stopped.

Adapting this task to use machine learning entailed replacing the hard-coded threshold parameter with a machine learning algorithm. Through supervised training the robot should learn to match the appropriate turn direction with a given sensor value.
A side effect of machine learning for this task is increased flexibility. The robot should be able to learn to follow the left edge of the line just as easily as the right. This capability was demonstrated during algorithm testing. Theoretically, the robot should also be able to learn line following on different colored lines and backgrounds.

4.0 Algorithm Selection

The online perceptron algorithm was a good fit for this machine learning task for two reasons. First, the algorithm is very simple mathematically, so it conforms well to the limitations of the Lego Mindstorms programming environment. Few variables and only basic mathematical operations are required. Since algorithm updates require only addition or subtraction, even the limitations of integer math don't present a problem. Secondly, the online perceptron algorithm lends itself well to interactive demonstration. Because the algorithm performs learning at each step, the correct answers can be given to the machine interactively by an operator. This allows the demonstration to be designed such that students can take part in actively "teaching" the task to the robot.

The online perceptron algorithm, as implemented for this project, is shown below. Note that the use of inequalities in evaluating g differs slightly from the algorithm presented in class. This modification was used to conform to the standard conditional functions in Robolab.

$$y^i \in \{1,-1\}, \quad h^i(\theta)= g(\theta T x^i), \quad g = \begin{cases} 1, & \theta T x^i > 0 \\ -1, & \theta T x^i \le 0 \end{cases}$$

$$\text{if } h^i(\theta) \neq y^i, \quad \theta := \theta + y^i x^i$$

5.0 Implementation

5.1 Robot Construction & Modification

For simplicity, the standard Terrabot design from the Lego Mindstorms for Schools kit was selected as the robotic base. The robot was then outfitted with sensors as shown in Figure 3. One light sensor was installed on the front of the vehicle for line detection. Two touch sensors were mounted facing upward to serve as operator inputs.

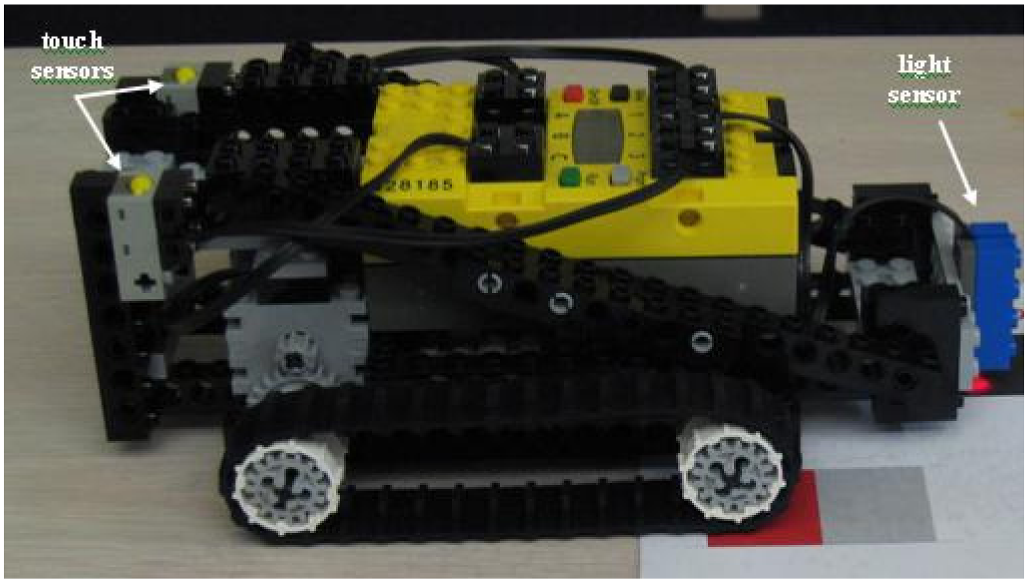

Figure 3: Lego Mindstorms Terrabot

5.2 Algorithm Implementation

The Robolab implementation of the online perceptron algorithm is shown in Figure FIXME. In this implementation, x is one-dimensional, so θ has two terms. These are represented by the red and blue containers. In each step, x is read in from the light sensor and is stored in the yellow container.

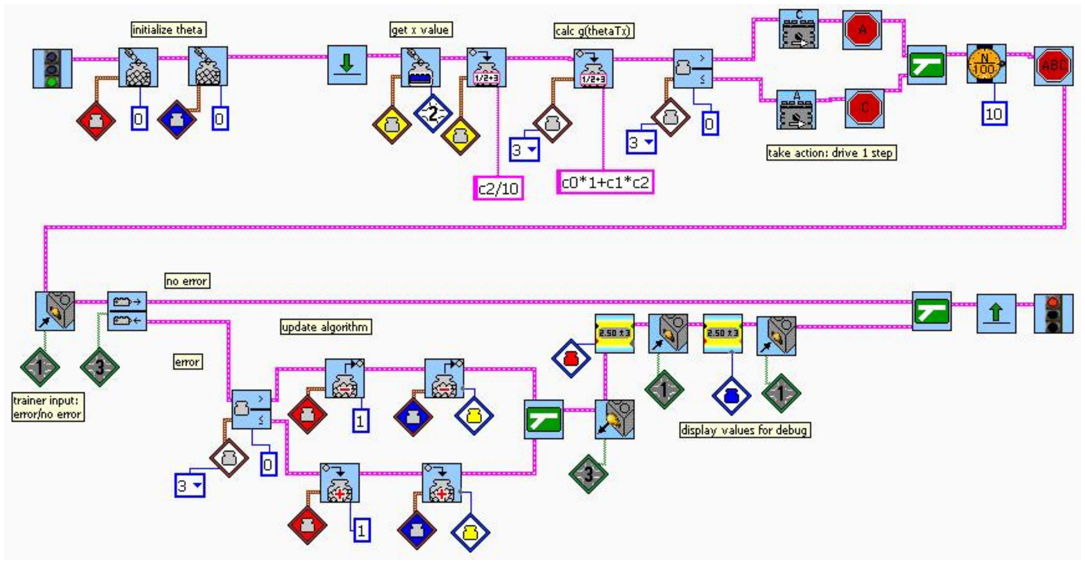

Figure 4: Online Perceptron algorithm implemented in Robolab

The first fork in the diagram evaluates $g(\theta^T x)$ and selects a turn direction. The robot takes one small step and then waits for training input from the operator. If the operator input indicates an incorrect turn, θ is updated. After the update, the new values are presented on the robot display for debugging purposes, and another step is initiated.

5.3 Training Process

To perform training, the robot is placed near the edge of the line, and the program is started. The robot takes a step and waits for input. If the robot correctly turns toward the edge of the line, the left touch sensor is pressed to proceed to the next step. If the robot turns away, the right touch sensor is held down to indicate an error while the left touch sensor is pressed to proceed.

6.0 Results

Training tests were performed using the black line on the activity pad included with the Lego Mindstorms for Schools kit, shown in Figure 1.

Initial tests were not successful. After dozens of trips up and down the training line, the robot had not learned the task. Investigation revealed that the x ranged from 39 over the dark line to 50 over the white paper. When using the online perceptron algorithm, the bound on training errors is proportional to D^2, where D is the maximum magnitude of x. The cause of the difficulty was exposed. With D = 50, hundreds, if not thousands, of errors could be expected.

To improve training performance, the output of the light sensor was divided by 10 before being stored as x for calculations. This reduced the magnitude of x to 3-5, a factor of 10 reduction in D, and a factor of 100 reduction in training error bound. This change proved to be very effective, and subsequent training tests succeeded.

A set of 15 tests were run, as recorded in Table 1. All 15 tests were successful. The robot was trained to follow the right or left edge of the black line, and the learning performance was not significantly affected by the side of the line used for training. The number of training errors varied between 18 and 34, with an average of 22.3 and standard deviation of 4.4.

Table 1: Training Results

Test #	Edge	# Errors
1	Right	22
2	Right	18
3	Right	20
4	Right	20
5	Right	20
6	Right	21
7	Right	30
8	Right	20
9	Right	20
10	Right	34
11	Left	22
12	Left	18
13	Left	24
14	Left	24
15	Left	22

The observed level of performance is sufficient to allow use of this project as a grade-school demonstration of machine learning. Training the robot takes only a few minutes, and the results are satisfying and entertaining. Perhaps this project will inspire some of the next generation of roboticists and computer scientists.