A Robust Wall-Following Robot That Learns by Example

Cameron Hassall, Rohan Bhargava, and Thomas Trappenberg

Dalhousie University, Halifax, NS B3H 3J5, Canada

Abstract. We designed and implemented a wall-following robot that learned appropriate actions from a small set of noisy training examples. We then assessed the effectiveness of this strategy by physically modifying the robot prior to training. Our system used nearest-neighbour classification to make motor command predictions, but we also tested two regression techniques for continuously varying action selection: linear regression and support vector regression (SVR). We found both classification and regression to be satisfactory prediction strategies for this task, although the advantage of SVR for predicting a continuum of actions became clear when the number of possible class labels was reduced.

1 Introduction

A robot rarely has access to a full and accurate description of its environment, and must therefore often act under uncertainty (Russell and Norvig, 2003). This potential uncertainty extends to a robot's physical characteristics which, in real-world applications, may change in profound ways. Motors and limbs may become damaged or destroyed, sensors can become occluded, and components may be reassembled incorrectly following repair. For example, a robot may be unable to make accurate predictions about its position after issuing a specific motor command. In other words, it lacks knowledge of the plant equation, the equation that specifies the effect of executed actions. We are interested in finding an effective strategy for a robot to recover after its physical configuration has been changed.

Schulz et al. (2009) proposed a design for a self-adapting robotic arm that was robust to damage and modification. Their system learned by trying out different motor commands, then observing the effects using an external camera. Given a desired position, their robot used a nearest-neighbour method to select appropriate motor commands. This technique required no a priori model knowledge. In many ways, a damaged or modified robot should be seen as a new, untrained robot, since the old plant equation may no longer be valid.

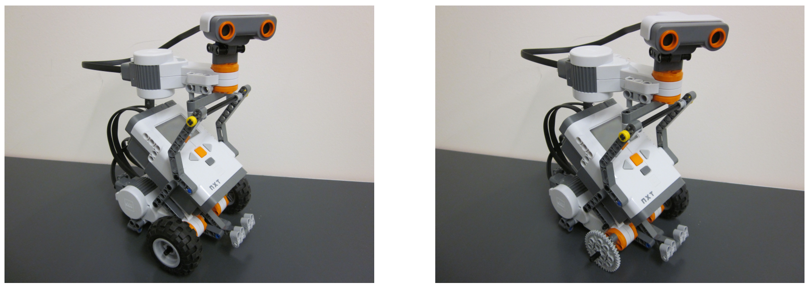
Here we chose a simple wall-following task that had enough complexity to represent a nonlinear control problem. The wall-following task requires that a robot travel at a set distance from a wall, or within a pre-specified margin. Freire et al. (2009) tested different neural network classifiers to predict the best actions for this task based on current sensor data. They found that even this seemingly-simple navigation task required complex decision-making. To simplify the problem, they reduced the number of inputs to the networks (distances, in their case).

In our study we wanted to compare the effectiveness of different strategies in solving the adaptive wall-following task with a small number of very noisy training data. We were specifically interested in comparing nearest-neighbour classification (Schulz et al., 2009) with regression methods able to predict a continuum of possible actions. We therefore evaluated how the number of classes influenced performance, and tested simple linear regression and the more sophisticated SVR. The robot was built from standard Lego Mindstorms NXT components and only the very inaccurate ultrasonic sensor was used to sense the environment. Our aim was to compare simple supervised learning strategies where the robot was shown examples of appropriate behaviour from which it had to generalize. To make sure that this learning could accommodate different plants, we tested several alterations of the robot, including changing one wheel to a different size, and interchanging the wheel motor control cables.

2 Design

2.1 Hardware

Mindstorms NXT by Lego is a flexible robotics kit that includes multiple sensors and actuators. The robot used in this study was a standard NXT tribot, along with a top-mounted ultrasonic sensor on a motor that provided 360 degrees of rotation (Figure 1). A front-mounted active light sensor was available, but unused in our design. All three motors included an internal rotational sensor.



(a) Unmodified Robot (b) Modified Robot

Fig. 1: NXT Tribot with pivoting ultrasonic sensor, before and after modification. Modifications included reducing the size of one wheel, and swapping the motor control cables.

2.2 Software

RWTH - Mindstorms NXT Toolbox is a MATLAB toolbox that allows remote control of the NXT system via a Bluetooth or USB connection (RWTH Aachen University, Germany, 2008).The MATLAB NXT Toolbox was used for all robot control and data collection. Data analysis and visualization were also done in MATLAB. SVR was done using LIBSVM, a MATLAB library for support vector machines (Chang and Lin, 2011).

2.3 Design Strategy

Internal Representation One of the simplest ways to have a robot perform a task is to show it exactly what to do, from start to finish. Specifically, we tried recording a sequence of motor commands (i.e. changes in position for the left and right motors). The motor commands were then executed, in order. This strategy only used an internal representation - it did not rely on external sensors or feedback. This robot learned to follow a specific path, and was not a general learning machine. In no way did it learn to generalize to new situations. Furthermore, there was enough noise in the motor sensors and movements that a large positional error tended to accumulate over time. This was not an effective or interesting strategy.

Feedback Controller The use of negative feedback is a common control strategy (van Turennout, Honderd, and van Schelven, 1992). We tested a manually-tuned proportional controller (P controller). We found this strategy to be unstable and not robust to modification. We observed that following many test modifications, the feedback controller could not be properly tuned to compensate.

Supervised Learning The goal of supervised learning is to generalize from training examples to new situations. In order to create a general and robust wall-follower, we trained our robot by showing it what to do in a given state. States were defined as vectors of distances, measured at various angles perpendicular to the wall (-30, -9, 0 9, 40, 60, 90) using the ultrasonic sensor.

$$\boldsymbol{d} = (d_1 d_2 ... d_7) \tag{1}$$

This gave our robot inputs similar to a coarse sonar. We observed noise in the ultrasonic sensor data, especially at greater distances and for detecting certain surface types.

Following this measurement, the tribot was manually moved to the desired position, and the change in motor positions was recorded by reading the motor positional sensors,

$$\boldsymbol{m^t} = (m_1^t \text{ and } m_2^t) \;\; , \tag{2}$$

and then comparing the new motor positions to the old:

$$m_1 = m_1^t - m_1^{t-1}, m_2 = m_2^t - m_2^{t-1} \tag{3}$$

For classification, training labels were manually assigned to each training example: left (L), slight left (SL), straight (S), slight right (SR), and right (R). Following training, average motor positions for each command were computed, e.g.

$$\boldsymbol{m}_{LEFT} = average(m_1, m_2) \text{ such that } label(m_1, m_2) = L \tag{4}$$

During the wall-following task, new ultrasonic data were recorded and compared to the training data. The new data were then classified using the k-nearest neighbour (k-NN) algorithm (Russell and Norvig, 2003). This algorithm works by classifying new observations based on the k-closest training examples, by majority vote. In our case, we chose k = 5, which seemed to result in fewer testing errors (misclassifications) compared to k = 3, or higher ks. During initial testing, we noticed that invalid ultrasonic readings of 255 cm were common and unwelcome. Manhattan distance, or 1-norm, was chosen over Euclidean and higher-order norms in order to minimize the effect of large differences between components.

$$distance\,(\boldsymbol{d_1} - \boldsymbol{d_2}) = \sum_i |d_{1i} - d_{2i}| \tag{5}$$

Following classification of the current state, the system simply applied the corresponding motor command as computed by averaging over the training data. Thus, each training label became associated with a single set of motor commands. Figure 2 gives an overview of the complete system.

Schulz et al. (2009) interpolated between nearest-neighbours to continuously vary motor commands for a robotic arm. We tested a similar strategy by applying two regressions to our training data - one for each motor value. Furthermore, we tested two different regression techniques: linear regression, and support vector regression. Our goal for this testing was for the robot to make continuously-varying and highly-appropriate motor command predictions. We felt that this would result in a more flexible system in contrast to a classifier because it would be capable of producing a wider range of movements.

3 Results

The goal of this project was to design a robot that could be trained to follow a wall within a distance of three feet, even after the robot had been physically modified. We recorded 25 training examples with training labels. These examples were recorded following modification. A 5-NN classifier was used for action selection. Presented here are the results of several modifications on wall-following performance, and a comparison of classification and regression techniques.

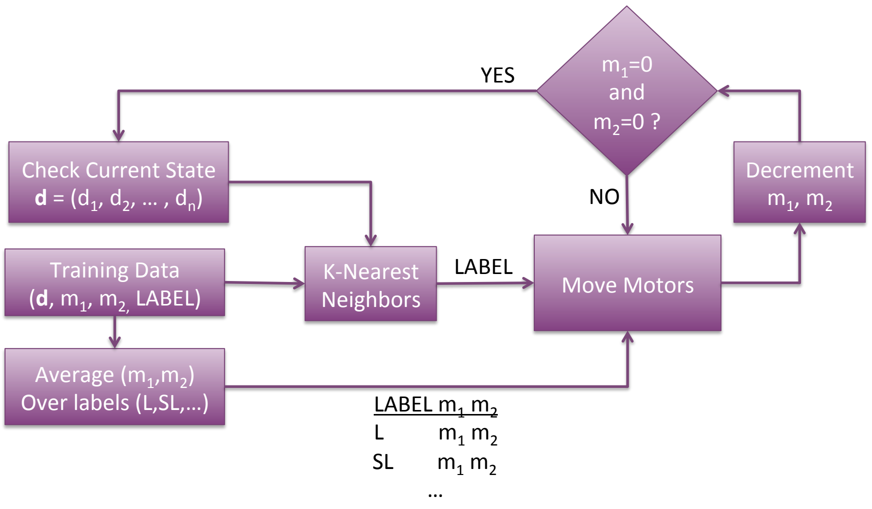

YES
m₁=0 and m₂=0 ?
Check Current State d = (d1, d2, … , dn)
NO
Decrement m1, m2
Training Data (d, m1, m2, LABEL)
K-Nearest Neighbors
LABEL
Move Motors
Average (m1,m2) Over labels (L,SL,…)
LABEL m1 m2
L m1 m2
SL m1 m2
…

Fig. 2: Design overview. The current state is compared to the training states, and k-NN is used to select an appropriate action, based on averaging the training results. Each motor is then moved until m_1 and m_2 are reached.

3.1 Modifications

One Wheel Replaced In this condition, one wheel was replaced with a smaller wheel (Figure 1). If the replacement wheel provided good traction, the robot could be retrained to complete the task correctly. Otherwise, we observed a mismatch between what was trained, and what actually occurred under the robot's own power. In other words, the wheel would slip. Turns became slower, but were not impossible. Straight-line travel was still possible, but awkward because we held the power constant in our design. This meant that motor commands that were unequal did not finish at the same time. "Straight" movements were thus split into two angled movements.

Both Wheels Replaced Replacing both wheels with smaller ones resulted in slower overall performance as compared to the unmodified robot. Again, if the wheels lacked traction then there seemed to be little correlation between the actions we showed the robot and what it could actually do on its own, and both wheels would often slip. With proper traction, though, the robot could be properly retrained. It should be noted that in this case the original unmodified robot training data was still somewhat valid, since the motor commands were symmetrical in both cases.

Motor Wires Swapped Swapping the motor wires had no effect on our robot's performance. During training, we labelled the examples based on how the robot was actually moved. For example, a left turn didn't rely on which motor changed (left or right). Critically, the change in both motors was recorded, and then turned into a generic motor command that reflected what was done in training, not some preset rule about how turns work.

3.2 Regression

As Figure 3 suggests, both classification and regression can make valid motor command predictions using only a few noisy data points. Among the regression techniques, SVR appeared to make slightly better predictions compared to linear regression for at least two out of the five sample data points (”Slight Right” and ”Slight Left”). What is interesting is that if fewer classes were used (e.g. ”Left”, ”Straight”, and ”Right” only) then SVR proved to be more accurate than nearest-neighbour classification. This suggests that regression, and in particular SVR, may be a better prediction strategy than classification if the training labels do not properly cover the range of desired outputs.

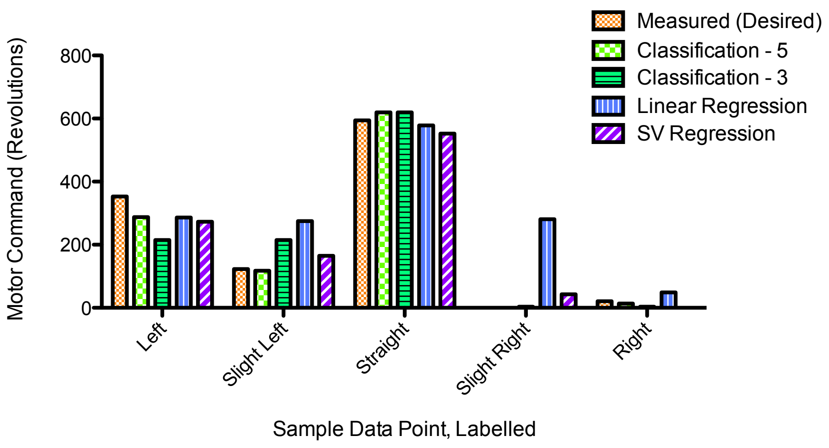


Fig. 3: Comparison of Classification and Regression. Predicted motor commands are shown for 5 sample data points for one motor. Included as a benchmark are the desired motor commands for each point. Support vector regression predictions were close to the desired commands more often than the linear regression predictions. Both regression techniques tended to predict reasonable commands. If fewer training labels were used (e.g. 3 instead of 5) then support vector regression tended to make better predictions than classification.

4 Conclusions

We hypothesized that a robot that learns how to move from training examples should be robust to modification and damage, as long as new training examples were recorded. Out results supported this "learning by example" strategy in the sense that our system could still perform the wall-following task after modification, although it could not always perform it in the same amount of time. This is reasonable, since any modification to a tested and effective design would likely result in a performance cost. What is important is that our system generally appeared to choose appropriate actions given the current state, and that the learned actions compensated for the modifications.

These results are consistent with the general finding of Schulz et al. (2009) that learning by examples is a robust strategy. Unlike their system, though, we did not include a path-planning component. Our system simply chose actions based on training examples. It reacted to the current state in the way it was shown during training. Also unlike Schulze et al. (2009), we found k-NN to be an effective action- selection strategy, and we did not find it necessary to interpolate between neighbours (although we did average over training labels to come up with motor commands). A likely explanation for this is that the density of our sample space (states and labels) was quite low compared to the Schulz et al. (2009) robotic arm, which tried to capture all possible arm positions and inputs for six motors.

The main limitation in this study was noise in the ultrasonic sensor readings. Our strategy depended on the distinguishability of states. Discriminability was good, but not perfect. Choosing a higher k-value for the k-NN classification seemed to account for this noise, although we did observe some unusual actions from our robot, such as taking a slight left or right when a straight would have been more appropriate. Luckily, the robot seemed to self-correct in these situations.

We mainly treated this as a classification problem, with five discrete actions. Regression testing gave favourable results, and in the future we would like to implement this learning technique to continuously vary the motor commands in response to the current state. Based on our results, this would be especially beneficial over a classification technique if the the training classes failed to properly capture the range of desired robot actions. We hypothesized that regression-based learning would be most advantageous for a continuously-moving robot. Our robot had to remain stationary in order to read the current state, due to design constraints.

We might also incorporate a feedback component, although this would introduce the problem of how to adapt the feedback controller to the modification. Finally, to address the shortcomings of the ultrasonic sensor, we propose using a more accurate laser rangefinder. It would be interesting to see if improved sensor accuracy would result in better state recognition, or if that was not the limiting factor in this study

References

Chang, C, Lin, C.: LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

Freire, A. L., Barreto, G. A., Veloso, M., Varela, A. T.: Short-Term Memory Mechanisms in Neural Network Learning of Robot Navigation Tasks: A Case Study. Robotics Symposium (LARS) (2009) 1-6

Russell, S. J., Norvig, P.: Artificial Intelligence: A Modern Approach. New Jersey: Prentice Hall (2003)

RWTH Aachen University, Germany (2008). RWTH - Mindstorms NXT Toolbox. URL: http://www.mindstorms.rwth-aachen.de

Schulz, H., Ott, L., Sturm, J., Burgard, W.: Learning Kinematics from Direct Self-Observation using Nearest-Neighbor Methods . Advances in Robotics Research (2009) 11-20

van Turennout, P., Honderd, G., van Schelven, L.: Wall-following Control of a Mobile Robot. Proceedings of the 1992 IEEE International Conference on Robotics and Automation (1992) 280-285

Yata, T., Kleeman, L., Yuta, S.: Wall Following Using Angle Information Measured by a Single Ultrasonic Transducer. Proceedings of the 1998 IEEE Conference on Robotics and Automation (1998) 1590-1596