# Learning to Avoid Collisions

**Elizabeth Sklar**[1,4], **Simon Parsons**[1,4], **Susan L. Epstein**[2,4], **A. Tuna Özgelen**[4], **J. Pablo Muñoz**[4], **Farah Abbasi**[3], **Eric Schneider**[2] and **Michael Costantino**[3]
[1]Brooklyn College, [2]Hunter College, [3]College of Staten Island, [4]The Graduate Center
The City University of New York
New York, NY USA
*contact author: sklar@sci.brooklyn.cuny.edu*

## Abstract

Members of a multi-robot team that operates within close quarters must avoid collisions. The typical, simple collision avoidance method has the robot compute its distance to other robots and stop, even move away, when this distance falls below a fixed threshold. While this approach may skirt disaster, it may also reduce the team's efficiency, when robots halt at length for others to pass by, or travel further to move around one another. This paper reports on experiments where a human trainer, through a graphical user interface, watches robots perform an exploration task. The trainer can direct the robots to halt before they collide, and then to resume once their paths are clear. Experiment logs record the robots' states, and a classifier is learned to identify the states in which "halt" and "resume" commands are issued. Preliminary results indicate that it is possible to learn a classifier that models trainers well, and that different trainers consider different factors when making their decisions.

## Introduction

We are interested in the use of human-robot teams to solve problems that are dangerous for all-human teams, but beyond the capabilities of all-robot teams. Examples of such tasks include *urban search and rescue* (Jacoff, Messina, and Evans 2000; Murphy, Casper, and Micire 2001) and *humanitarian de-mining* (Habib 2007; Santana, Barata, and Correia 2007). In urban search and rescue, robots explore an enclosed space, such as a collapsed building, and seek to locate human victims. In humanitarian de-mining, robots explore an open space, such as a field in a war zone, to search for anti-personnel mines that may be concealed. The goal is to locate mines so that they can be disarmed and the region rendered safe.

In both cases, teams of robots are deployed to locate targets of interest in terrain that is potentially unsafe for people, and in both cases the robots will typically need a person to help with parts of the task that they cannot easily handle on their own. In urban search and rescue, this might be identifying a human victim, and in de-mining, determining what kind of device the robot team has located.

Our work (Sklar et al. 2011; 2012), focuses on inexpensive, limited-function robots. We believe that teams of such robots are more practical for wider deployment on our targeted tasks than teams of fewer, more expensive and more capable robots. Although individual robots may require human assistance, large teams of robots present considerable challenges for human-robot interaction. There are, however, several ways that robots can profitably learn from a human trainer, and perhaps lessen the needs for human assistance in some circumstances. It is also necessary to consider the similarities and differences among human trainers' strategies, to support the eventual construction and incorporation of a high-performing collision avoidance mechanism that models human decision making.

This paper reports on a feasibility study of one such instance, where a person intervenes to avert collisions. Our preliminary investigation collected data from 5 human subjects and assessed the ability to learn something useful from it. The results, detailed here are promising and warrant further investigation.



## Related Work

The idea that an interactive system can improve its behavior through observation of human users' key strokes and mouse clicks (*data mining the clickstream*) is not new. In the 1960's and 1970's, Teitelman developed an automatic error correction facility that grew into DWIM (Do What I Mean) (Teitelman 1979). In the early 1990's, Cypher created Eager, an agent that learned to recognize repetitive tasks in an email application and take them over from the user (Cypher 1991). Maes used machine learning techniques to train agents to help with email, filter news messages, and recommend entertainment. These agents gradually gained confidence in their understanding of users' preferences (Maes 1994).

In robotics, the idea that robots can learn from humans has been explored as *learning by demonstration* (Argall et al. 2009), also known as *programming by demonstration* (Hersch et al. 2008). This is commonly viewed as a form of reinforcement learning, which learns a policy from a sequence of state/action pairs (Argall et al. 2009). Other approaches to robots learning from people include (Katagami and Yamada 2000), where human teachers provide examples that seed evolutionary learning; (Lockerd and Breazeal 2004), where the robot tries to identify a human's goal and make its own plan to achieve it; and (Nicolescu and Matarić 2001), where the robot observes as the human execites actions in

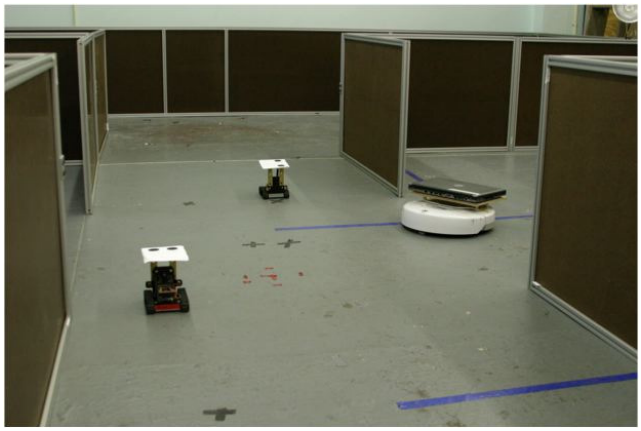


Figure 1: The robots' physical environment.

its domain, and learns the outcomes of its own actions from these observations. Little of this work is concerned with multiple robots, however. There is a long history of multi-robot learning, for example (Matarić 1997; Parker 2000; Bowling and Veloso 2003; Pugh and Martinoli 2006), but it involves learning from trial and error, not learning from a human teacher.

In earlier related work, we collected moves during the play of video and educational games, and used the data to train neural networks that then served as controllers for opponents in the same games (Sklar 2000; Sklar, Blair, and Pollack 2001). The aim was not to produce the best player, but rather to derive a population of players that represented different characteristics of play. This technique was later extended beyond games to generate populations of agents that emulated students performing at different skill levels on an educational assessment (Sklar et al. 2007; Sklar and Icke 2009). Here, once again, we apply this method to capture and then represent characteristic behaviors of different people as they interact with a large human-robot team performing a complex task.

## Our Approach

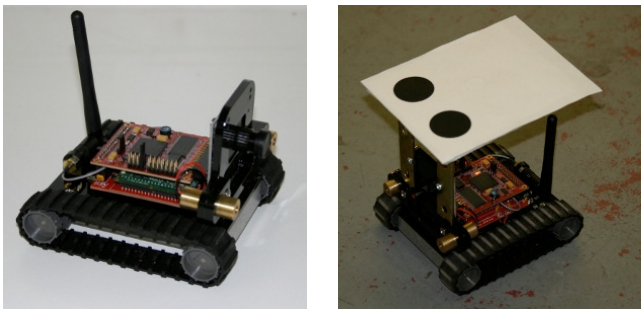
The work reported here involved a single human trainer who interacted with a team of three robots. This section describes the physical environment in which the experiments were conducted and the experimental design.

### Physical setup

Our experimental testbed models the interior of a building, with a large space of six rooms and a hallway, which the robots explore, as described below. The physical testbed is shown in Figure 1. The full space is approximately 400 square feet .

The robots used for these experiments are Surveyor SRV-1 Blackfins. The Blackfin[1] is a small tracked platform equipped with a webcam and 802.11 wireless. The Blackfin

[1] http://www.surveyor.com/SRV_info.html



(a) unmodified (b) with hat

Figure 2: The Surveyor SRV-1 Blackfin.

is pictured in Figure 2(a). Localization is provided by a network of overhead cameras (Sklar et al. 2011).To help these cameras distinguish among otherwise identical robots, each robot is provided with a unique "hat." A Blackfin wearing a hat is shown in Figure 2(b). (Each has has a letter from the Braille alphabet, one without rotational symmetry so that the hat provides orientation as well as position.)

Because the Blackfin has limited on-board processing, the controller for these robots runs off-board, and communicates with the robot over a wireless network, which naturally results in some lag. The controller for all the robots on the team, plus software to allocate tasks to robots, and the software that extracts robot positions from the overhead cameras, runs on a group of machines that make up the control station for the experiments, located next to the arena. The control station is pictured in Figure 3.

### Motivation

As explained above, we want our human/robot team to explore the physical space. Here we continue experiments where robots are allocated particular *interest points* that they must visit. As described in (Sklar et al. 2012), a market-based component allocates these points to the robots on a team. The robot controllers then plot a path that covers

Figure 3: The control station.

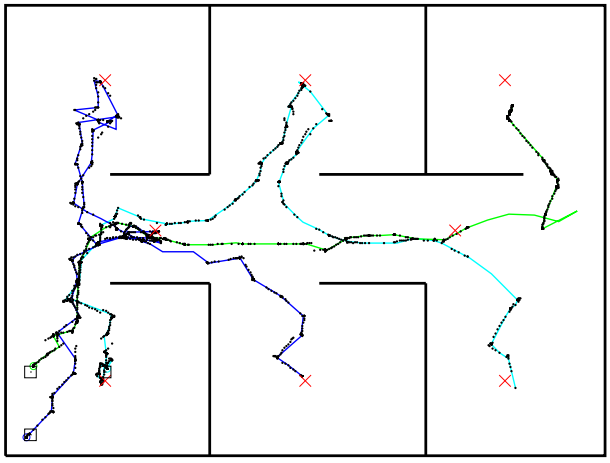


Figure 4: Paths traced by three robots exploring the arena. The robots start in the lower left "room," at the points marked with black □'s. Each robot visits an assigned subset pf the 8 "interest points" indicated by red ×'s.

all the points that their robots have been allocated, with no knowledge of what other robots are planning to do. Then the robots simultaneously maneuver to their assigned points, in accordance with their plans.

Several experimental factors place the robots in each other's way. First, they are in a restricted space, Second, in some experimental configurations they all begin in the same part of the space, to model a situation where they all enter the space from the same point. Finally, each robot has no knowledge of what the other robots are planning to do, This mutual interference is clear in Figure 4, which shows the motion of three robots during one experimental run. Because of this interference, the robot controller is programmed to prevent collisions, and it does so very conservatively — it halts any robots that get too close to one another. The robot that is closest to its target interest point is then given right-of-way, while the other(s) wait until it is no longer in their vicinity; then they are given right-of-way to move again. The amount of time the robots wait is recorded as *delay time*, which can accrue rapidly if many robots are trying to maneuver in the same small space.

This *fixed-threshold method* is a simplistic way to prevent collisions. It is reasonable to ask if a person could provide a good model of how avoid collisions in a way that reduces overall delay time. The next section describes our initial iattempt to learn such behaviors from people.

### Experimental setup

The robot team began in the configuration of Figure 4, that is, tin the lower-left room, and were allocated eight interest points, as described in (Sklar et al. 2012). The robots then started to follow the paths that they had planned.

The conservative collision avoidance mechanism described above was disabled. Instead, collision avoidance was in the hands of a human subject, the *trainer*. This person sat at an "operator station" physically remote from the control station and the arena, in a separate room in the lab complex. The trainer could not see or hear anything from the arena or the control station. The only information that the trainer had about the robots appeared on a user interface, show in Figure 5. For each robot, the interface displayed its current position and orientation and its next interest point, Robot position and orientation are derived by the overhead cameras, and are therefore subject to error and lag.

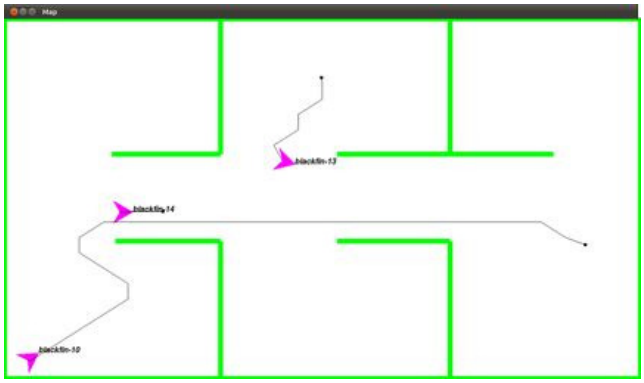


Figure 5: The user interface presented to the human trainer.

The trainer could send two commands to the robots from her computer keyboard: wait or resume movement. To send either command to a single robot, the trainer first clicked on the robot's icon to select it, and then clicked on the command. To send either command to all the robots, the trainer merely clicked on it.

Each experiment consisted of a single run of the system. The robots were positioned, interest points were allocated, and the robots then maneuvered to those points while the human trainer monitored for collisions. In each run, the trainer made the robots wait when she thought it necessary to avoid a collision, and then restarted them when she judged that the danger of collision was past. A run ended when the robots all reached their final interest point or when a robot collided with another robot or a wall. Operators at the control station set up the robots, ran the task allocation mechanism, and detected any collisions.

## Experiments and Results

The results reported here were obtained five human subjects, each of whom was responsible for five runs, and were undergraduate researchers working in our lab (2 female, 3 male). All the trainers had previously participated in similar experiments, and therefore required no training on the interface. We restricted data to runs where no robot collided with a wall; each trainer worked until she had completed five such runs. Each run ended either with a robot-robot collision or with successful completion of the full task.

During each run, the system logged data continually, including the current positions of the robots and each robot's path to its target location (i.e., sequence of waypoints). Each time the human trainer clicked on a robot to stop its move-

ment (presumably to avoid a collision with another robot), a "Wait" command was logged. Similarly,whenever the trainer clicked (again) on a robot to resume its movement, a "Resume" command was logged. Thus, the log file for each run can be used to reconstruct the world state for each robot, at each point in time; and these states can be labeled with the human trainer's decision to perform an action or to do nothing (i.e., let the system perform autonomously). A subject robot's state is represented as:

$$\langle r_1, \theta_1, r_2, \theta_2, V_x, V_y, H_x, H_y \rangle \quad (1)$$

where $r_1, \theta_1$ and $r_2, \theta_2$ are the range (in cm) and angle (in radians), respectively, from the subject robot to the other two robots in the arena; $V_x$ and $V_y$ are the $x$ and $y$ velocities of the subject robot, and $H_x$ and $H_y$ are the heading of the subject robot (i.e., the $x$ and $y$ distance from its current location to the next waypoint in its planned path). Distances are computed as straight-line Euclidean distances, without consideration of intervening walls or other robots or obstacles.

We analyze the collected data to address three questions: (1) the ability to learn from log data, (2) the ability to distinguish between trainers, and (3) the difference between the trainers' methods and the fixed-threshold method for collision avoidance.

## Ability to learn from log data

To address the first question, whether the log files could support learning to avoid collisions, we first coded the log data. Table 1 describes the data for the five human trainers, listed as H1 through H5. The first column indicates the total number of states described in the log files for each trainer. The next four columns show the number of instances labeled in each of four categories:

- moving-moving indicates that the robot was moving when the message was logged
- moving-waiting indicates that the human trainer clicked on the robot to stop its movement, so that a "Wait" command was logged
- waiting-moving indicates that the human trainer clicked on the robot to resume its movement, so that a "Resume" command was logged
- waiting-waiting indicates that the robot was not moving when the message was logged.

We want to learn which states, as represented in Equation 1, belong to which of the four labels. The upper portion of the table, however, clearly indicates that the vast majority of instances are moving-moving. This imbalance has a serious negative impact on learning: most classifiers will simply default to the moving-moving label. (Understandably so; the selection of the majority class will make them correct 99.72% of the time.) A robot guided by such a classifier would never wait. To mitigate this, we created a balanced training set, as follows. We retained all the non-moving-moving instances, and then extracted equally many moving-moving instances across the data set, at equally-spaced intervals in the chronologically-ordered log file. Statistics on the balanced training set appear in the lower portion of Table 1.

We tested 10 classifiers as implemented in WEKA[2] (version 3.6.7) (Witten, Frank, and Hall 2011) with their default parameters and 10-fold cross validation: $k$-nearest neighbors, C4.5 decision trees, a rule extractor from a decision tree (PART), naïve Bayes, Holte's OneR rule learner, a support vector machine (SMO), logistic regression, AdaBoost, logit boost, and decision stumps. The appendix summarizes the Weka default values.

To learn to classify an instance, we ran each classifier on the balanced subset data. Table 2 shows the accuracy of the best classifier for each trainer. None is much better than random .

| | | |
|---|---|---|
| H1 | rule learner | 54.69% |
| H2 | support vector machine | 50.00% |
| H3 | C4.5 decision tree | 59.34% |
| H4 | logistic regression | 56.16% |
| H5 | rule learner | 56.80% |

Table 2: The best results on learning 4 classes from the balanced subset data.

The resultant confusion matrices clearly indicated that the moving-waiting and waiting-moving labels were not reliably distinguished. Table 3 shows the confusion matrix for the best result on 4 classes, the balanced subset data for trainer H3, (corresponding to the third row in Table 2). Clearly moving-waiting and waiting-moving are confused with each other more often than either is classified correctly. Inspection of the states themselves indicated that, indeed, the conditions under which the human trainers choose to halt a robot's movement because it was about to collide with something were almost identical to the conditions under which the trainer determined that it was safe for a halted robot to begin moving again.

| *classified as* → | moving-moving | moving-waiting | waiting-moving |
|---|---|---|---|
| moving-moving | 38 | 4 | 3 |
| moving-waiting | 5 | 13 | 5 |
| waiting-moving | 5 | 15 | 3 |

Table 3: Confusion matrix for the best result learning 4 classes from the balanced subset data for H3. This training set did not contain any instances of waiting-waiting because H3 never invoked that state.

This observation led us to relabel the data, this time into two classes. The moving-moving and waiting-waiting instances were relabeled as no-action instances, and the moving-waiting and waiting-moving instances were relabeled as action instances. The last two columns of Table 1 show the number of training instances after relabeling. Again, learning was conducted on both the complete set of training data, and the balanced subset data. Table 4 shows

[2] http://www.cs.waikato.ac.nz/~ml/weka

| | *num. of samples* | moving-moving | moving-waiting | waiting-moving | waiting-waiting | no-action | action |
|---|---|---|---|---|---|---|---|
| *complete set of training data* | | | | | | | |
| H1 | 20,800 | 20,768 | 16 | 16 | 0 | 20,768 | 32 |
| H2 | 17,103 | 17,058 | 23 | 22 | 0 | 17,058 | 45 |
| H3 | 22,503 | 22,457 | 23 | 23 | 0 | 22,457 | 46 |
| H4 | 15,221 | 15,148 | 35 | 35 | 3 | 15,151 | 70 |
| H5 | 17,451 | 17,389 | 31 | 31 | 0 | 17,389 | 62 |
| *balanced subset of training data* | | | | | | | |
| H1 | 65 | 32 | 16 | 16 | 0 | 32 | 32 |
| H2 | 91 | 45 | 23 | 22 | 0 | 45 | 45 |
| H3 | 92 | 45 | 23 | 23 | 0 | 45 | 46 |
| H4 | 146 | 72 | 35 | 35 | 3 | 70 | 70 |
| H5 | 125 | 62 | 31 | 31 | 0 | 62 | 62 |

Table 1: Training data. Each $Hi$ indicates one of the 5 human trainers.

the best results on the balanced subset data. They represent a considerable improvement over Table 2.

| | | |
|---|---|---|
| H1 | k-nearest neighbor | 90.63% |
| H2 | logit boost | 76.67% |
| H3 | logit boost | 87.91% |
| H4 | AdaBoost | 82.86% |
| H5 | k-nearest neighbor | 87.10% |

Table 4: The best stratified 10-fold cross validation results, learning 2 classes from the balanced subset data.

Table 5 shows the confusion matrix for the best of these, on the balanced subset data for trainer H1. Again, this is a substantial improvement over the results shown in Table 3. Current work includes the investigation of other techniques to address imbalanced data, that would allow us to use a larger sample of the data.

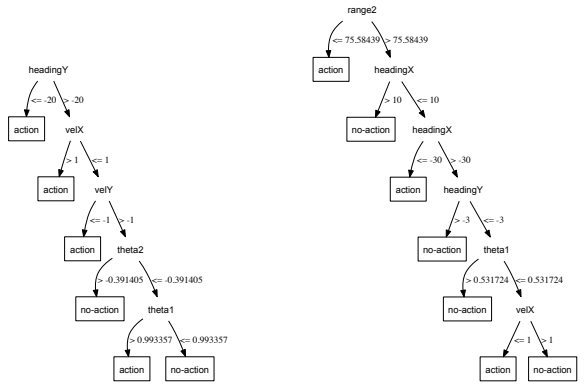
| *classified as* → | no-action | action |
|---|---|---|
| no-action | 30 | 2 |
| action | 4 | 28 |

Table 5: Confusion matrix for the best result learning 2 classes from the balanced subset data (H1).

## Ability to distinguish between trainers

The second question, whether the data could be used to differentiate between trainers, is relatively simple and straightforward. Decision trees that model the individual trainers clarify the differences among them. Figure 6 shows the C4.5 tree learned for the two most accurately modeled trainers.

It is interesting to note that the decision trees for different trainers depend upon different aspects of the robot's world state. The decisions of H1, as captured by the decision tree, depend on the direction that the robot of interest is heading (headingY, its speed (velX ($V_x$) and velY ($V_y$)), and the directions that the other two robots are heading (theta1 ($\theta_1$)




H1 (85.94%) H3 (86.81%)

Figure 6: Decision Trees (C4.5) for the two most accurately modeled trainers. The percentage of correctly identified instances appears in parenthesis.

and theta2 ($\theta_2$)). In contrast, the decisions of H3 depend its distance from one of the other robots (range2 ($r_1$)), that robot's heading (headingX ($H_x$) and headingY ($H_y$)), its speed (velX ($V_x$)) and the direction of the third robot (theta1 ($\theta_1$)), not the same robot as the one whose distance is considered. H1 appears to focus on trajectories alone, while H3 attends first to proximity.

We also analyzed the differences in behaviors identified by the decision stump results. Table 6 shows the rules obtained using this method for the five trainers. Again, it is clear that the decisions of different humans, as extracted by the decision stump algorithm, are influenced by different components of the robot's world state.

While these observations are preliminary, and we have not yet collected enough data from human subjects to produce definitive models of particular individuals' behaviors, these results do tell us that our direction is promising. Current work includes the collection of more data from human sub-

| | | |
|---|---|---|
| H1 | (68.75%) | *act if you are heading not too far to the south* |
| H2 | (66.67%) | *act if you are going quickly west* |
| H3 | (71.43%) | *act if you are closer than 77cm to one of the other robots* |
| H4 | (75.00%) | *act if you are heading not too far to the east* |
| H5 | (78.23%) | *act if you are closer than 124cm to one of the other robots* |

Table 6: Decision Stump rules for all five trainers, with the percentage of correctly classified in instances in parenthesis.

jects, programming the decision trees and decision stump rules into the robot controllers, and evaluation of the system's performance under the strategies modeled on different human subjects.

### Comparison to the fixed-threshold method for collision avoidance

The third question addressed here is how collision avoidance decisions made by human subjects compare to the fixed-threshold method employed by our current system. Our intuition and initial hypothesis was that the human trainers would allow robots to get closer to each other than the fixed-threshold method would. As a result, the amount of time that a robot spent waiting for others to move out of the way would decrease. The fixed-threshold method used a distance of 50cm, with results reported in (Sklar et al. 2012).

Interestingly, the human trainers who made decisions based on distances between robots (i.e., either $r_1$ or $r_2$) used more than 50 cm: 77cm and 124cm in the decision stump rules for trainers H3 and H5, respectively; and 75.59cm, 61.07cm and 74.85cm in the decision trees for H3, H4 and H5, respectively. The other trainers, however, did not consider the distance to other robots at all. Instead, they examined other factors, such as the directions in which the robots were heading. In the decision trees, where more complex decision rules can be coded, even those trainers who did consider distance also considered the directions in which the robots were heading. So our preliminary conclusion is that the fixed-threshold method does not take into consideration all the important factors in collision avoidance. Our current work includes an analysis of the delay time for all the experiments described here.

## Summary

This paper reports preliminary results from experiments in which a human guides a team of robots to avoid collisions. We recorded data from human subjects instructed to help members of a robot team to avoid collisions with one another. Subjects could instruct the robots to wait or resume moving, but could not steer them. Based on data recorded while the human subjects were engaged in this task, we trained a classifier to predict when the robots should change state (toggle from moving to waiting or from waiting to moving). The best-fitting classifier achieved 90% accuracy. A more rigorous evaluation will use such a classifier to control the robots without a human trainer, and gauge its ability to avoid collisions. Ideally, one would learn from the best-performing human trainers. Our next major step is to identify such skilled people, and learn from data we collect as they efficiently defend against robot collision.

## Acknowledgments

This work was supported by the National Science Foundation under grants #IIS-1117000 and #CNS-0851901.

## Appendix: WEKA defaults

weka.classifiers.trees.J48 (C4.5): confidence threhold for pruning = 0.25; minimum number of instances per leaf = 2; seed for random data shuffling = 1.

weka.classifiers.lazy.IBk (k-nn): nearest neighbour search algorithm = weka.core.neighboursearch.LinearNNSearch; number of nearest neighbours ($k$) used in classification = 1.

weka.classifiers.rules.PART: confidence threhold for pruning = 0.25; minimum number of instances per leaf = 2; seed for random data shuffling = 1.

weka.classifiers.bayes.NaiveBayes (naïve Bayes without kernel): use normal distribution for numeric attributes.

weka.classifiers.rules.OneR (OneR): minimum number of objects in a bucket = 6.

weka.classifiers.functions.SMO (svm): kernel = weka.classifiers.functions.supportVector.PolyKernel; complexity constant $C$ = 1; normalize; epsilon for round-off error = $1.0e-12$; use training data for internal cross-validation; random number seed = 1; size of the cache = 250007; exponent = 1.0; do not use lower-order terms.

weka.classifiers.functions.Logistic (logistic regression): maximum number of iterations = $-1$ (until convergence).

weka.classifiers.meta.AdaBoostM1 (AdaBoost): percentage of weight mass to base training on = 100; random number seed = 1; number of iterations = 10; base classifier = weka.classifiers.trees.DecisionStump.

weka.classifiers.meta.LogitBoost (logit boost): percentage of weight mass to base training on = 100; number of folds for internal cross-validation = 0 (no cross-validation); number of runs for internal cross-validation = 1; threshold on the improvement of the likelihood = $-\text{Double.MAX\_VALUE}$; shrinkage parameter = 1; random number seed = 1; number of iterations = 10; base classifier = weka.classifiers.trees.DecisionStump.

## References

Argall, B.; Chernova, S.; Browning, B.; and Veloso, M. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Blair, A. D., and Sklar, E. I. 1999. Exploring evolutionary learning in a simulated hockey environment. In *Proceedings of the 1999 Congress on Evolutionary Computation*, 197–203.

Blair, A. D.; Sklar, E. I.; and Funes, P. 1998. Co-evolution, determinism and robustness. In *Simulated Evolution and Learning*, volume 1585 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag. 389–396.

Bowling, M., and Veloso, M. 2003. Simultaneous adversarial multi-robot learning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.

Cypher, A. 1991. Eager: Programming repetitive tasks by example. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*.

Funes, P.; Sklar, E. I.; Juillée, H.; and Pollack, J. B. 1988. Animal-animat coevolution: Using the animal population as fitness function. In *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, 525–533. MIT Press.

Habib, M. K. 2007. Humanitarian Demining: Reality and the Challenge of Technology. *Interational Journal of Advanced Robotic Systems* 4(2):151–172.

Hersch, M.; Guenter, F.; Calinon, S.; and Billard, A. 2008. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*.

Jacoff, A.; Messina, E.; and Evans, J. 2000. A standard test course for urban search and rescue robots. In *Proceedings of PerMIS*.

Katagami, D., and Yamada, S. 2000. Interactive classifier system for real robot learning. In *IEEE International Workshop on Robot and Human Interaction*, 258–263.

Lockerd, A., and Breazeal, C. 2004. Tutelage and socially guided robot learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7):31–40.

Matarić, M. 1997. Reinforcement learning in the multi-robot domain. *Autonomous Robots* 4:73–83.

Murphy, R. R.; Casper, J.; and Micire, M. 2001. Potential tasks and research issues for mobile robots in RoboCup Rescue. In *Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Artificial Intelligence*. Springer.

Nicolescu, M. N., and Matarić, M. J. 2001. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics* 31(5):419–430.

Parker, L. 2000. Multi-robot learning in a cooperative observation task. In *Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems*.

Pugh, J., and Martinoli, A. 2006. Multi-robot learning with particle swarm optimization. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*.

Santana, P. F.; Barata, J.; and Correia, L. 2007. Sustainable Robots for Humanitarian Demining. *International Journal of Advanced Robotic Systems* 4(2):207–218.

Sklar, E. I., and Icke, I. 2009. Using simulation to evaluate data-driven agents. In *Multi-agent Based Simulation IX*, volume 5269 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Sklar, E. I.; Salvit, J.; Camacho, C.; Liu, W.; and Andrewlevich, V. 2007. An agent-based methodology for analyzing and visualizing educational assessment data. In *Proceeding of the Sixth International Conference on Autonomous Agents and Multiagent Systems*.

Sklar, E. I.; Özgelen, A. T.; Muñoz, J. P.; Gonzalez, J.; Manashirov, M.; Epstein, S. L.; and Parsons, S. 2011. Designing the HRTeam framework: Lessons learned from a rough-'n-ready human/multi-robot team. In *Proceedings of the Workshop on Autonomous Robots and Multirobot Systems*.

Sklar, E. I.; Özgelen, A. T.; Schneider, E.; Costantino, M.; Muñoz, J. P.; Epstein, S. L.; and Parsons, S. 2012. On transfer from multiagent to multi-robot systems. In *Proceedings of the Workshop on Autonomous Robots and Multirobot Systems*.

Sklar, E. I.; Blair, A. D.; and Pollack, J. B. 2001. Training intelligent agents using human data collected on the internet. In *Agent Engineering*. Singapore: World Scientific. 201–226.

Sklar, E. I. 2000. *CEL: A Framework for Enabling an Internet Learning Community*. Ph.D. Dissertation, Department of Computer Science, Brandeis University.

Teitelman, W. 1979. A display oriented programmer's assistant. *International Journal of Man-Machine Studies* 11:157–187.

Witten, I. H.; Frank, E.; and Hall, M. A. 2011. *Data Mining, Practical Machine Learning Tools and Techniques*. Elsevier Inc., third edition.