The efficiency of the RULES-4 classification learning algorithm in predicting the density of agents

Ziad Salem1* and Thomas Schmickl2

Abstract: Learning is the act of obtaining new or modifying existing knowledge, behaviours, skills or preferences. The ability to learn is found in humans, other organisms and some machines. Learning is always based on some sort of observations or data such as examples, direct experience or instruction. This paper presents a classification algorithm to learn the density of agents in an arena based on the measurements of six proximity sensors of a combined actuator sensor units (CASUs). Rules are presented that were induced by the learning algorithm that was trained with data-sets based on the CASU's sensor data streams collected during a number of experiments with “Bristlebots (agents) in the arena (environment)”. It was found that a set of rules generated by the learning algorithm is able to predict the number of bristlebots in the arena based on the CASU’s sensor readings with satisfying accuracy.

1. Introduction

Inductive learning is the process of creating a hypothesis or a classifier from examples that can be generalized and applied to new examples (Mitchell, 1997). Generally, a classifier can make two types of classification errors in new examples for a two-class problem. It can misclassify positive instances as negative and negative instances as positive. The rate of correct predictions made by the classifier is the prediction accuracy of this classifier in the specific data-set (Kotsiantis, Pierrakeas, & Pintelas, 2004).

ABOUT THE AUTHOR

Ziad Salem is an associate professor, Computer Engineering Department, Electrical and Electronic Engineering faculty, Aleppo University, Aleppo, Syrian Arab Republic. He completed his BSc in the year 1991. He completed his PhD in the area of machine learning and data mining from the Intelligence Research Laboratories, Systems Engineering Division, Cardiff School of Engineering, Cardiff University, Cardiff, UK in the year 2002. He published over 30 journal and conference papers, and has over 10 years of teaching experience at national and international universities. He reviewed tenth of journals and conference papers from different fields. He participated in four different European projects. His research interest falls in the area of artificial intelligence, data mining, machine learning, knowledge discovery, optimum solution and swarm intelligence.

PUBLIC INTEREST STATEMENT

The paper describes the training of a classification algorithm on a live sensor data stream produced by six proximity sensors of a combined actuator sensor unit (CASU). The purpose of the investigation described in this paper was to assess the ability of the learning algorithm to determine the number of honeybees or bristlebots in an arena equipped with a CASU. We placed different numbers of bristlebots in the arena and analysed the rules generated by the algorithm with regard to their ability to discriminate between low and high bristlebot densities.
Learning tasks can be very different from each other in many aspects, including the kind of input or feedback they get (examples or examples with hints), the way they are tested (during the learning process or only at the end, number of allowed mistakes) and the level of control over the learning process (does it affect the world or just observe it?). Thus, many different models have been suggested for formalizing the learning process (Murthy, 1998; Quinlan, 1986).

In rule induction systems, a decision rule is defined as a sequence of Boolean clauses linked by logical AND operators that together imply membership in a particular class (Fürnkranz, 1999). The general goal is to construct the smallest rule set that is consistent with the training data-set. A large number of learned rules are usually a sign that the learning algorithm tries to “remember” the training data-set instead of discovering the assumptions that govern it (Fürnkranz, 1997). During classification, the left-hand sides of the rules are applied sequentially until one of them evaluates to true, and then the implied class label from the right-hand side of the rule is offered as the class prediction (Cleetus & Dhanya, 2014).

The type of learning described in this paper is known as supervised machine learning, which designates the search for algorithms that reason from externally supplied instances (examples) to produce general hypotheses, which then make predictions about future instances (Quinlan, 1993). The goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown (Kotsiantis, 2007).

Every example in any data-set used by machine learning algorithms is represented using the same set of features. The features may be continuous or discrete. If examples are given with known labels (the corresponding correct outputs), then the learning is called supervised, in contrast to unsupervised learning, where examples are unlabelled. By applying these unsupervised (clustering) algorithms, researchers hope to discover unknown, but useful, classes of items (Jain, Murty, & Flynn, 1999).

On the other hand, zoologists and psychologists study learning in animals and humans. There are several parallels between learning in animals and machine learning where many machine learning techniques are derived from the efforts of psychologists to make more precise their theories of animal and human learning through computational models. Thus, the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning (Nilsson, 1998).

Concept learning is a central part of animal cognition that enables appropriate motor response in novel situations by generalization of former experience, possibly from a few examples (Sandin et al., 2014). Michalski claimed that to satisfy hunger, an animal must be able to classify some objects as edible despite the great variety of their forms and changes they undergo in the environment (Michalski, 1983). Thus, an intelligent system must be able to form concepts, that is, classes of entities united by some principle that might be a common use or goal (Michalski, 1986).

The work described in this paper is concerned with classification problems in which the output (the class) assumes only discrete, unordered values that represent the number of the bristlebots in the arena, while the attribute values are continuous.

This work is part of the Animal and Robot Societies Self-organise and Integrate by Social Interaction—bees and fish (ASSISIbf) project, which generates a mixed society of honeybees and artificial (robotic) agents as a novel bio-hybrid system that can achieve self-awareness, self-regulation and environmental awareness through self-organization and collective information processing (Schmickl, Szopek, & Bodí, 2013). The main goal of ASSISIbf is to establish a robotic society that is able to develop communication channels with animal societies (honeybees and fish swarms)
on its own (Schmickl, Bogdan, et al., 2013). The scientific goals of ASSISI are to develop robots that can influence the collective behaviours of animals (bees and fish), establish an adaptive and self-organizing society built by robots and animals, enable the robots to autonomously learn the social language of the animals, establish mixed societies that pursue a common goal which can be defined by human users of the system and allow the robots to gain novel skills by incorporating the capabilities of the animals (sensors and cognition).

2. Learning process
Choosing and using a specific learning algorithm is a critical step. Once preliminary testing has produced satisfying results, the classifier (mapping from unlabelled instances to classes) is available for routine use. The evaluation of the classifier is most often based on prediction accuracy which is the proportion of correct predictions with regard to the total number of predictions (Kotsiantis, 2007).

There are many ways to calculate a classifier’s accuracy. One way is to split the data-set, using two-thirds for training and one-third for estimating the performance (unseen data-set). For another method, known as cross-validation, the training set is divided into mutually exclusive and equal-sized subsets and for each subset, the classifier is trained on the union of all the other subsets. The average of the error rates of each subset is therefore an estimate of the error rate of the classifier. This validation is computationally expensive but useful when the most accurate estimate of a classifier’s error rate is required. If the error rate is too high, a variety of factors has to be examined (Vanitha & Niraimathi, 2013). Features that are relevant for the problem might be ignored by the classifier, a larger training might be needed, the dimensionality of the problem might be too high, parameters extracted from the data-sets might need to be retuned and the selected algorithm might be inappropriate. High error rates can also occur as a consequence of imbalanced data-sets (Japkowicz & Stephen, 2002).

3. Brief description of the machine learning algorithm
Machine learning studies automatic techniques for making accurate predictions or choosing useful actions based on experience or observations (Alpaydin, 2010; Murphy, 2012).

Many successful applications and algorithms of machine learning exist already.

The choice of the machine learning algorithm to be used was important as the algorithm will be implemented in the combined actuator sensor units (CASUs) in the future. Since these devices will not provide much computational power, the selection of the most simple algorithm suitable for solving the given problem was a crucial decision. For this reason, we opted for the RULES-4 algorithm (Pham & Dimov, 1997), which is a widely used incremental inductive learning algorithm from the “RULES” family of automatic rule extraction systems. It allows the stored knowledge to be updated and refined rapidly when new examples are available. RULES-4 shares some common features with its immediate predecessor, RULES-3 Plus (Pham & Dimov, 1996) such as the rules forming procedure as shown in Figure 1. The RULES-4 algorithm is summarized in Figure 2. The algorithm extracts rules incrementally by processing one example at a time.

Figure 3 explains the interaction between short-term memory (STM) and long-term memory (LTM) in the algorithm. The learning process iterates over the examples of the training data-set and loads a new example into the STM in every step. When the STM is full, a random example is discarded from the STM to make room for a new example. The final rule set produced by the algorithm is kept in the LTM for evaluation with the test data-set.

The STM and LTM of the algorithm works in analogy with the organization of biological memory. The STM works similarly to the human STM as it stores a very limited amount of information (e.g. a phone number). This information is processed by the brain and the results may be transferred to LTM where it may be stored for a long time or removed after not being used for a longer period.
Figure 1. Rule forming procedure of RULES-3 Plus and RULES-4.

| Step | Description |
|------|-------------|
| 1.   | Take an unclassified example and form array SETAV (SET of Attributes and Values). |
| 2.   | Initialise arrays PRSET and T_PRSET (PRSET and T_PRSET will consist of \( m_{PRSET} \) expressions with null conditions and zero \( H \) measures) and set \( n_{CO} = 0 \). |
| 3.   | IF \( n_{CO} < n_{A} \) THEN \( n_{CO} = n_{CO} + 1 \) and set \( m = 0 \); ELSE the example itself is taken as a rule and stop. |
| 4.   | DO |
|      | \( m = m + 1 \); |
|      | Specialise expression \( m \) in PRSET by appending to it a condition from SETAV that differs from the conditions already included in the expression; |
|      | Compute the \( H \) measure for the expression; |
|      | IF its \( H \) measure is higher than the \( H \) measure of any expression in T_PRSET THEN replace the expression having the lowest \( H \) measure with the newly formed expression; |
|      | ELSE discard the new expression; |
|      | WHILE \( m <= m_{PRSET} \) |
| 5.   | IF there are consistent expressions in T_PRSET THEN choose as a rule the expression that has the highest \( H \) measure and discard the others; |
|      | ELSE copy T_PRSET into PRSET; |
|      | Initialise T_PRSET and go to Step 4. |

Where, SETAV – SET of Attributes and Values; PRSET – Partial Rules SET; T_PRSET – a temporary array of partial rules of the same dimension as PRSET; \( m_{PRSET} \) – number of expressions stored in PRSET (\( m_{PRSET} \) is user-provided); \( n_{CO} \) – number of conditions; \( n_{A} \) – number of attributes; \( m \) – number of expressions.

Figure 2. Incremental induction procedure in RULES-4.

| Input | Short-Term Memory (STM), Long-Term Memory (LTM), ranges of values of the numerical attributes, frequency distribution of examples among classes, one new example. |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Output| Short-Term Memory (STM), Long-Term Memory (LTM), updated ranges of values of the numerical attributes, updated frequency distribution of examples among classes. |

| Step 1 | Update the frequency distribution of examples among classes. |
| Step 2 | Test whether the values of the numerical attributes are within their existing ranges and if not update the ranges. |
| Step 3 | Test whether there are rules in the LTM that classify or misclassify the new example and simultaneously update their accuracy measures (\( A \) measures) and \( H \) measures. |
| Step 4 | Prune the LTM by removing the rules for which the \( A \) measure is lower than a given prespecified level (threshold). |
| Step 5 | IF the number of examples in the STM is less than a prespecified limit THEN add the example to the STM ELSE |
|      | IF there are no rules in the LTM that classify the new example THEN replace an example from the STM that belongs to the class with the largest number of representatives in the STM by the new example. |
| Step 6 | For each example in the STM that is not covered by the LTM, form a new rule by applying the rule forming procedure summarised in Fig. 1 of RULES-3 Plus. Add the rule to the LTM. Repeat this step till there are no examples uncovered by the LTM. |
Figure 4 shows a representation of the mind’s operations, which is similar to the algorithm flowchart (Figure 3) where the contents in the STM are permanently lost when all the training data-set examples have been trained. On the other side, the LTM is refined every time a new example enters the STM until reaching the final stage where permanent rule sets are generated (stage analogous to the permanent memory stores in Figure 4).

Thus, repeatedly occurring events are stored in permanent memory comparable to rules being stored in the rule set, which will then be used to test the strength of the algorithm based on the unseen data-set.

A similar scenario has been developed by Hunt (1982) and Sinclair (2010, 2011) with a system of buffers that enables humans to organize incoming stimuli, a LTM of essentially infinite capacity, and a short-term or working memory that does the human conscious mental work.

Pham and Dimov (1997) state that the input information needed to start the induction process in the RULES-4 algorithm includes:

(a) A set of examples examined in some of the previous iterations and retained in the STM. The size of the memory has to be predefined by the user. Initially, the STM is empty.

(b) The set of rules extracted so far and stored in the LTM. At the beginning, the LTM can be empty or can contain initial rules defined by the user. Apart from the rules themselves, statistical information is stored regarding the numbers of examples correctly classified and misclassified by each rule in the LTM. Using this information, a decision is made as to which rules to prune from the LTM.
(c) Ranges of values for the numerical attributes. As with RULES-3 Plus, the RULES-4 algorithm can also deal with numerical attributes by quantizing them. The ranges of values for these attributes are updated before processing every new example. The number of quantization levels for each range is specified by the user. For the experiments presented in this paper, the quantization level has been set to six levels. When a new example is processed, RULES-4 forms a corresponding description in which the values of all numerical attributes are represented by appropriate quantization levels. Induction is then carried out with the new description, the quantization levels being treated as any other values.

(d) The number of examples found in each class so far.

It is very important for a rule induction algorithm to generate decision rules that have high predictability, which is normally measured by a function called rule quality. This function is needed in both the rule induction and classification processes such as $H$-measure, which has been used in the RULES-3 algorithm (Pham & Dimov, 1996). $H$-measure for a given expression is defined in Equation 1, as:

$$H = \sqrt{\frac{E^c}{E}} \times \left[2 - 2 \sqrt{\frac{E^i}{E}} \frac{E_j}{E} - 2 \sqrt{1 - \frac{E^i}{E}} \left(1 - \frac{E_j}{E}\right)\right]$$

(1)

where $E^c$ is the number of examples covered by the expression (the total number of examples correctly classified and misclassified by a given rule), $E$ is the total number of examples, $E^i$ is the number of examples covered by the expression and belonging to the target class $i$ (the number of examples correctly classified by a given rule) and $E_j$ is the number of examples in the training set belonging to the target class $i$. In Equation 1, the first term

$$G = \sqrt{\frac{E^c}{E}}$$

(2)

relates to the generality of the rule and the second term,

$$A = 2 - 2 \sqrt{\frac{E^i}{E}} \frac{E_j}{E} - 2 \sqrt{1 - \frac{E^i}{E}} \left(1 - \frac{E_j}{E}\right)$$

(3)

relates to its accuracy (Lee, 1994).

The important question to ask, when dealing with machine learning algorithms, is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem.

The objective is to utilize the strengths of one method to compensate for the weaknesses of another. If interested in the best possible classification accuracy, it would be difficult to find a single classifier that reaches a performance similar to that of a good group of classifiers. There is a trade-off between classification accuracy and the computational effort to increase accuracy without decreasing comprehensibility (Guyon & Elisseeff, 2003), where the point to which the classification accuracy needs to be optimized depends on the type of application.

4. Experimental set-up
This section describes the set-up used in the experiments described in this article and introduces a new and unique application for the learning algorithm.

4.1. The combined actuator sensor unit
The CASUs are static robots with the ability to produce heat and vibration (actuators) and to sense nearby objects with the help of six infrared driven proximity sensors. A set of homogeneous CASUs distributed in a grid-like configuration over an arena shared with animals that respond
to heat or vibration can be employed to provide the animals with an interactive environment. Figure 5 shows a picture of the CASU used in this research. The idea behind the integration of the CASUs into animal societies is to adapt the behaviour of the CASUs to influence the behaviour of the animals, for example, to make young honeybees aggregate at a number of target points designated by heat or vibration or to make them choose between targets of different quality. In order to achieve the desired behaviours in the animal societies, evolutionary algorithms will be used as adaptation mechanisms to produce appropriate decentralized controllers for the CASUs.

Figure 6 shows the mechanical design of a CASU. The CASU body consists of a lower metal part and an upper plastic part, which is the only part visible to bees. The hexagonal upper part of the CASU is equipped with six infrared (IR) proximity sensors that guarantee an all-round coverage for detecting honeybees up to a distance of 20 mm (diameter of the CASU). These sensors do not influence the bee’s behaviour since they do not see IR light (Menzel, 1977). In the experiments described in this paper, we used the six proximity sensors to detect the bristlebots in the arena.

4.2. The bristlebot

The bristlebot is one of the simplest forms of mobile robots, both in function and in construction. It is a rigid-bodied robot, the bottom of which is equipped with bristles, like the head of a brush. The majority of the bristles are inclined against the vertical, which gives the bristlebot a preferred “forward” direction when its body is vibrated and the bristles convey the vibration to the ground, thus propelling the bristlebot. Most bristlebots are powered electrically, making use of modern developments in low-mass motors and batteries.

4.3. The arena

In this study, we used a round arena as shown in Figure 7 with a floor made of wax sheets (standard wax combs as used in bee keeping) and a plastic wall coated with Teflon in order to keep the bees from climbing the wall.
4.4. Honeybees
The honeybees (Apis mellifera ssp.) presented in this paper were extracted from their hives as sealed brood with their brood comb and were reared in an incubator until hatching. We used them for experiments during the first 24 h after hatching to exploit their inability to fly and their known temperature preference at this age. Their preference for specific temperatures is fundamental to their temperature-induced aggregation behaviour, one of several forms of collective behaviour (swarm intelligence) found in honeybees.
5. Data description and processing
The initial collection of the data is done by recording the proximity sensors’ values, which represent the intensities of the reflections of the pulsed infrared signals emitted by the IR sensors at a rate of 10 s⁻¹ to detect moving agents in the arena within a maximum range of 2.0 cm. Each data-set contains the actual number of bristlebots in the arena and 595 records collected during 59.5 s with six columns that represent the values of the six sensors of a single CASU. We opted to integrate the sensor readings of 1 min into a single prediction to achieve good classification accuracy at an acceptable rate. For evaluation, the sensor values were converted to the Boolean values 0 (no detection) and 1 (detection) based on a user-specified sensor-specific threshold which we set to 10% of the respective sensor’s average value in the data-set. We repeated the experiments 109 times with bristlebot densities varying between one and nine bots in the arena.

Table 1 shows a sample of the Boolean data derived from the sensor readings collected from the CASU’s proximity sensors with nine bristlebots in the arena.

In order to conclude from the sensor activity to the number of bristlebots in the arena, we defined a set of suitable continuous attributes that are based on the rate of activity (number of active states per number of measurements) of individual sensors and combinations of sensors. For example, one set of attributes is based on the number of concurrently active sensors. A sensor was considered as active whenever its sensor readout-value exceeded a pre-defined threshold θ.

Another set of attributes is based on the number of concurrently active neighbours (2–6), semi-neighbours (2 or 3) and opposite sensors.

Table 2 shows the sub-set of attributes used in the learning experiments along with the type of each attribute (continuous). Since three active semi-neighbours and active opposite sensors never occurred, they were not considered for this study and were thus excluded from the set of evaluated attributes.

Based on the sub-set of attributes, we created a new data-set with 109 examples, 9 attributes and 9 discrete classes (the number of bristlebots ranging from one to nine). For the learning process, the data-set was split into a training set (70 examples, 64%) and test set (39 examples, 36%) in these experiments.

| Table 1. A sample of the raw data for the recorded proximity sensor for nine bristlebots |
|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| Sensor 1 | Sensor 2 | Sensor 3 | Sensor 4 | Sensor 5 | Sensor 6 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
Table 3 shows the description of the data-set used in these experiments. Table 4 shows a sample of the data-set used in the learning experiments.

6. Experiments CASU-bristlebots
The purpose of the learning experiments is to assess the classification algorithm’s ability to classify the group size based on the live sensor data streams that are produced by the CASU proximity. The rules generated by the learning algorithm will enable the CASU to know how many honeybees or bristlebots are around it. Figure 8 shows an image of nine bristlebots in the arena with a diameter of 10 cm (Figure 8(A)), the detection ranges of the six sensors (Figure 8(B)) and the activity of one sensor during the course of a 1-min experiment (Figure 8(C)). Due to the availability of synchronously recorded videos, it is possible to synchronize the position of the bristlebots with the sensor readings they trigger.

Table 2. Used attributes and their values

| Attribute's name | Values       |
|------------------|--------------|
| one_active       | Continuous   |
| two_active       | Continuous   |
| three_active     | Continuous   |
| four_active      | Continuous   |
| two_ngb          | Continuous   |
| two_semngb       | Continuous   |
| two_opp          | Continuous   |
| three_ngb        | Continuous   |
| four_ngb         | Continuous   |

Table 3. A data-set description

| Total number of examples | Total number in training data-set | Total number in test data-set | Total number of attributes | Type of attributes | Total number of classes | Type of classes |
|--------------------------|----------------------------------|------------------------------|---------------------------|-------------------|------------------------|----------------|
| 109                      | 70                               | 39                           | 9                         | Continuous        | 9                      | Discrete        |

Table 4. A data-set sample

| n_bots | one_active | two_active | three_active | four_active | two_ngb | two_semngb | two_opp | three_ngb | four_ngb |
|--------|------------|------------|--------------|-------------|---------|------------|---------|-----------|---------|
| 1      | 0.0084     | 0.0134     | 0.0000       | 0.0000      | 0.0134  | 0.0000     | 0.0000  | 0.0000    | 0.0000  |
| 2      | 0.0118     | 0.0034     | 0.0000       | 0.0000      | 0.0034  | 0.0000     | 0.0000  | 0.0000    | 0.0000  |
| 3      | 0.0235     | 0.0252     | 0.0000       | 0.0000      | 0.0252  | 0.0000     | 0.0000  | 0.0000    | 0.0000  |
| 4      | 0.0739     | 0.0168     | 0.0000       | 0.0000      | 0.0168  | 0.0000     | 0.0000  | 0.0000    | 0.0000  |
| 5      | 0.0790     | 0.0303     | 0.0000       | 0.0000      | 0.0303  | 0.0000     | 0.0000  | 0.0000    | 0.0000  |
| 6      | 0.1059     | 0.0639     | 0.0000       | 0.0000      | 0.0588  | 0.0050     | 0.0000  | 0.0000    | 0.0000  |
| 7      | 0.2202     | 0.1143     | 0.0034       | 0.0034      | 0.1042  | 0.0050     | 0.0000  | 0.0000    | 0.0000  |
| 8      | 0.1714     | 0.1109     | 0.0336       | 0.0000      | 0.1076  | 0.0034     | 0.0000  | 0.0000    | 0.0218  |
| 9      | 0.2387     | 0.1666     | 0.0571       | 0.0134      | 0.1529  | 0.0101     | 0.0034  | 0.0043    | 0.0084  |
| 9      | 0.1664     | 0.1277     | 0.0084       | 0.0000      | 0.1143  | 0.0134     | 0.0000  | 0.0084    | 0.0000  |
Figure 9 shows three different scenarios of a CASU in the arena with different types of agents.

The first scenario (Figure 9(A)) shows honeybees in a larger arena \( (d = 60 \text{ cm}) \) with a simulated CASU that emulates the detections of a real CASU with the help of a visual tracking system to simulate CASU-based detection of honeybees in the area for later use. The second scenario (Figure 9(B)) shows a CASU with bristlebots in the arena. Figure 9(C) shows a CASU with honeybees in the arena. This paper concentrates on the second scenario of the CASU with bristlebot and describes how the classification algorithm would enable the CASUs to determine the number of bristlebots in the arena. The algorithm will be trained on the live sensor data stream(s) that are produced by the CASUs’ sensors to detecting the number of bristlebots around the CASU, from which it will be able to conclude to the number of bristlebots in the arena.

The goal of the experiments was to change the number of bristlebots in the arena and test whether the algorithm can successfully discriminate between low and high density of agents in the arena. For the further development of the CASUs, it is crucial to demonstrate their ability to learn on the fly how to evaluate the number of agents in the arena based on the data stream generated by its six proximity sensors.
The algorithm is designed to accept the maximum number of conditions in one rule and the maximum number of examples in the STM as user-specified parameters. These two parameters influence the accuracy and the number of generated rules and the coverage of all classes in the rule set, which is very important for the evaluation. The learning experiments were repeated 16,384 times and from the generated rule sets, we used those with the best classification accuracy after discarding all rule sets that did not cover all classes. The large number of repetitions is due to the nature of the incremental RULES-4 algorithm where a new hypothesis is generated in every run. Thus, the probability to find a good rule set grows with the number of runs.

Table 5 shows the results of running the algorithm on the collected data-set described in Table 3. The results of three conditions per rule are almost identical to the results of two conditions per rule; therefore, only experiments with one and two conditions were evaluated in this study. In addition, no accuracy improvement was observed for STM size of more than 15.

Figure 10 shows an exemplary rule set generated with one condition per rule.

Figures 11 and 12 show histograms of the frequencies of classification errors in the range [−9, +9] (−9 indicates an underestimation of 9 bristlebots, i.e. 9 bots were in the arena and none were detected and similarly +9 indicates an overestimation of 9 bots) made when applying the best rule set retrieved from the iterations of the algorithm with one and two conditions per rule.

The rule set with two conditions per rule is much better than that with one condition per rule when considering the number of correctly classified examples as shown in Figure 12. Therefore, the histogram of classification errors provides important information about the rule sets. For instance, the rule set with one condition per rule was able to correctly classify only 34.9% of the examples.

| Maximum number of conditions in a rule | Training results | Test results | Number of generated rules | STM size |
|---------------------------------------|-----------------|--------------|--------------------------|----------|
|                                       | Total number of examples covered by the rules | Classification accuracy (%) | Total number of examples covered by the rules | Classification accuracy (%) | |
| 1                                     | 69              | 0.463768     | 39                       | 0.384615 | 13   | 12 |
| 2                                     | 55              | 0.618182     | 26                       | 0.461538 | 24   | 15 |

**Figure 10. Generated rule set of one condition (STM = 12).**

| Rule |
|------|
| R1: 0.0100667 <= two_semg < 0.0117333 9 (C=1, A=1.36754, H=0.163453) |
| R2: 0.165867 <= one_active < 0.206483 9 (C=1, A=1.36754, H=0.163453) |
| R3: 0.14035 <= two_active < 0.1664 9 (C=1, A=1.36754, H=0.163453) |
| R4: 0.08825 <= two_active < 0.1143 7 (C=1, A=1.32388, H=0.158234) |
| R5: 0.00755 <= two_opp < 0.0100667 6 (C=1, A=1.32388, H=0.158234) |
| R6: 0.00284 <= four_rng < 0.0056 8 (C=1, A=1.32388, H=0.223776) |
| R7: 0.113167 <= two_active < 0.139783 9 (C=1, A=1.36754, H=0.231157) |
| R8: 0.00223333 <= two_semg < 0.0046667 7 (C=1, A=1.32388, H=0.158234) |
| R9: 0.0440167 <= one_active < 0.0846333 4 (C=1, A=1.32388, H=0.158234) |
| R10: 0.005 <= two_active < 0.0319 3 (C=1, A=1.32388, H=0.223776) |
| R11: 0.0846333 <= one_active < 0.12525 5 (C=0.333333, A=0.072791, H=0.0363955) |
| R12: 0.0034 <= two_active < 0.0305667 1 (C=0.5, A=0.211146, H=0.0861998) |
| R13: 0.0034 <= one_active < 0.0440167 2 (C=0.25, A=0.0318638, H=0.0183966) |

Where, C = rule consistancy, i.e. correct classifications per total classifications, A = rule accuracy, H = information content.
while the rule set with two conditions per rule could correctly classify 40.4% of the examples. However, when an absolute classification error of 1 can be tolerated, the rule set with one condition per rule performed better (72.5% acceptable classifications) than the rule set with two conditions (56% acceptable classifications). Therefore, the selection of the best rule set depends on the requirements of the task. The learning programme may be able to correctly classify only 50% of the instances, but it might be able to classify 90% with a maximum absolute error of 1, so that a still very good estimate for the number of robots would be available.

Increasing the STM size allows for more rules to be generated, and the larger rule sets achieve better results on the data-set the learning programme has been trained with.
Table 6 shows the average accuracies over all 16,384 experiments for different STM sizes for one condition (Table 6(A)) and two conditions (Table 6(B)). The two tables show that the average accuracy improves with growing STM size (Figure 13), which means higher STM sizes result in better rules being generated.

The correlation between STM size and classification accuracy is due to the fact that the rule learning process starts when the STM is full, so that a larger STM provides more examples to start learning from.

On the other hand, increasing the accuracy also leads to an increase in the number of examples classified correctly (prediction error = 0) as shown in Table 7. The table also shows the number of acceptable classifications, a measure which allows some fuzziness in the classification as it tolerates a certain classification error. In this case, we tolerated an error of one bot, so that the number of acceptable classifications is the number of correct classifications plus the number of underestimations by one (i.e. one bot too few) plus the number of overestimations by one (i.e. one bot too many).

Figure 14 shows a graph of the proportion of correct and acceptable classifications for rules with one condition, for which the best result is achieved in terms of both correct and acceptable classifications when STM size is 12.

| STM Size | Average accuracy |
|----------|------------------|
| 10       | 0.34370648       |
| 11       | 0.34978185       |
| 12       | 0.36166046       |
| 13       | 0.36619707       |
| 14       | 0.37130111       |
| 15       | 0.37327918       |

| STM Size | Average accuracy |
|----------|------------------|
| 10       | 0.4173521        |
| 11       | 0.43085665       |
| 12       | 0.45758678       |
| 13       | 0.46904649       |
| 14       | 0.4746033        |
| 15       | 0.49621111       |

Figure 13. The average accuracy for different STM sizes.
The results of two conditions rules are shown in Table 8 and Figure 15.

These results are promising as they prove that the rule sets generated by the learning algorithm are capable of predicting the number of agents in the arena with sufficient accuracy and that the learning algorithm itself can be implemented in the CASU to allow them to learn on the fly the number of the bristlebots in the arena.

### Table 7. Average accuracy results over 16,384 samples for one condition rules

| STM | $n_{\text{correct}}$ | $n_{\text{acceptable}}$ | Correct (%) | Acceptable (%) |
|-----|-----------------------|--------------------------|-------------|----------------|
| 12  | 38                    | 79                       | 34.9        | 72.5           |
| 11  | 28                    | 71                       | 25.7        | 65.1           |
| 10  | 29                    | 62                       | 26.6        | 56.9           |

### Table 8. Average accuracy results over 16,384 samples for two condition rules

| STM | $n_{\text{correct}}$ | $n_{\text{acceptable}}$ | Correct (%) | Acceptable (%) |
|-----|-----------------------|--------------------------|-------------|----------------|
| 10  | 42                    | 68                       | 38.5        | 62.4           |
| 11  | 32                    | 57                       | 29.4        | 62.3           |
| 12  | 39                    | 56                       | 35.8        | 51.4           |
| 13  | 47                    | 72                       | 43.1        | 66.1           |
| 14  | 43                    | 67                       | 39.4        | 61.5           |
| 15  | 44                    | 61                       | 40.4        | 56.0           |
7. Conclusions and future work

With the help of a simple machine learning algorithm, RULES-4, the density of bristlebots in the arena can be determined from the sensor readings of a few short-range proximity detectors. Although the algorithm is incremental, it can deal with both large and small data-sets, so that the quality of the generated rule sets does not depend on the size of the data-set. The algorithm can be used in the ASSISIbf project to allow a CASU to estimate the number of bees in the arena by implementing the results of the learning programme or even the learning algorithm itself in the software of the CASUs. It will also be interesting to use the optimized attributes and their thresholds in a simulation of virtual CASUs in recordings of the bee arena with known numbers of bees (i.e. an algorithm calculates the number of bees based on the attributes) and then evaluate the results to determine the quality of the predictions. In a first step, the best rule sets generated in this study can be implemented in the CASUs’ control software in order to optimize their ability to predict the number of honeybees in the arena.

The average accuracy of the learning algorithm could be improved by altering the algorithm’s user specified parameters.

The implementation of an on-the-fly learning system in a stationary robot that enables it to dynamically adapt its behaviour to the number of nearby agents is a novel approach that will provide a new level of interaction between stationary robots and mobile agents.

Future work will concentrate on experiments with honeybees. The learning algorithm will be used to determine whether a bee near the CASU is standing, moving slowly or fast, or rotating around the CASU (including information on the orientation). In order to accelerate our experiments with the CASUs, we will employ several arenas next to each other, each of which will be equipped with a CASU. This will enable us to run several experiments in parallel and thus increase the number of experimental runs per time.

Acknowledgements

I am grateful to my academic coordinator at Karl-Franzens-University Graz, Prof. Thomas Schmickl, who supplied me with invaluable advice and guidance throughout my time at the Artificial Life Lab concerning my research, writing, organization and life, and who supported me in my plans. Particularly, I am very grateful to Mr. Gerald Radspeier for his help and discussions. I am similarly grateful to Ms. Payam Zahadat, Mr. Ronald Thenius and Mr. Christoph Mosinger. Thanks to the ALL lab members. I am also very grateful for all the support I received from Prof. Stjepan Bogdan, head of the LARICS, Laboratory for Robotics and Intelligent Control Systems, department of Control and Computer Engineering, faculty of Electrical Engineering and Computing at the University of Zagreb and his team, Mr. Damjan Miklic, Mr. Tomislav Haus and Mr. Karlo Grigaric for their support in the software and hardware part of the experiment. I would like to acknowledge all the financial support I received during my research from Berlin technique university Prof. Olivier Pfeiffer, the Erasmus Mundus Project Coordinator (Averpace II. Jordan, Lebanon, Palestine and Syria), and Mr. Nils Wehmeyer, the project assistant. Finally, special thanks to the officials at Aleppo University and the ministry of higher education in Syria for participating in this project.

Funding

This work was supported by: EU-ICT “ASSISIbf”, [grant number 601074]; Austrian Science Fund (FWF) research [grant number P19478-B16], [grant number P23943-N13] (REBODIMENT), and Erasmus Mundus Partnership Action 2, Strand 1, Lot 5.

Author details

Ziad Salem1
E-mail: dr_ziad@hotmail.co.uk

Thomas Schmickl2
E-mail: thomas.schmickl@uni-graz.at

1 Electrical and Electronic Engineering Faculty, Computer Engineering Department, Aleppo University, Aleppo, Syria.
2 Artificial Life Lab of the Department of Zoology, Karl-Franzens University Graz, Universitätsplatz 2, 8010 Graz, Austria.

Citation information

Cite this article as: The efficiency of the RULES-4 classification learning algorithm in predicting the density of agents, Z. Salem & T. Schmickl, Cogent Engineering (2014), 1: 986262.

References

Alpaydin, E. (2010). Introduction to machine learning (2nd ed.). Cambridge, MA: MIT Press.

Cleetus, N., & Dhanya, K. A. (2014). Rule induction using ant-miner algorithm. International Journal of Scientific & Engineering Research, 5, 6–11.

Fürnkranz, J. (1997). Pruning algorithms for rule learning. Artificial Intelligence Review, 13, 3–54. http://dx.doi.org/10.1023/A:1006524209794

Fürnkranz, J. (1999). Separate-and-conquer rule learning. Artificial Intelligence Review, 13, 3–54. http://dx.doi.org/10.1023/A:100732942533

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3, 1157–1182.
Jain, A. K., Murty, M. N., & Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, 31, 264–323. [http://dx.doi.org/10.1145/331499.331504](http://dx.doi.org/10.1145/331499.331504)

Kotsiantis, S., Pierrakeas, C., & Pintelas, P. (2004). Predicting students' performance in distance learning using machine learning techniques. *Applied Intelligence*, 18, 411–426. [http://dx.doi.org/10.1023/B:AIAL.0000044797.32128.b5](http://dx.doi.org/10.1023/B:AIAL.0000044797.32128.b5)

Lee, C. (1994). Generating classification rules from databases. In Proceedings of the Ninth Conference on Application of Artificial Intelligence in Engineering (pp. 205–212). Malvern, PA, USA.

Menzel, R. (1977). Farbensehen bei Insekten—ein rezeptorphysiologischer und neurophysiologischer Problemkreis: Color vision in insects. *Verhandlungen der Deutschen Zoologischen Gesellschaft*, 1977, 26–40.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 83–134). Tioga: Morgan Kaufmann.

Michalski, R. S. (1986). Concept learning. In *Al encyclopaedia* (Vol. 1, pp. 185–194). New York, NY: Wiley.

Mitchell, T. M. (1997). Machine learning. New York, NY: McGraw-Hill. ISBN: 9780070428077; ISBN-10: 0070428077.

Murphy, K. P. (2012). Machine learning: A probabilistic perspective. Cambridge, MA: MIT Press.

Murphy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2, 345–389. [http://dx.doi.org/10.1023/A:100974630224](http://dx.doi.org/10.1023/A:100974630224)

Nilsson, N. J. (1998). *Introduction to machine learning*. An early draft of a proposed textbook. Stanford, CA: Robotics Laboratories, Department of Computer Science, Stanford University.

Pham, D. T., & Dimov, S. S. (1996). An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition*, 30, 1137–1143.

Pham, D. T., & Dimov, S. S. (1997). An algorithm for incremental inductive learning. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 211, 239–249. [http://dx.doi.org/10.1243/0954405971516239](http://dx.doi.org/10.1243/0954405971516239)

Quinlan, J. R. (1986). Induction of decision trees. *Journal of Machine Learning*, 1, 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco, CA: Morgan Kaufmann.

Sandin, F., Khan, A. I., Dyer, A. G., Amin, A. H. N., Indiveri, G., Chicca, E., & Osipov, E. (2014). Concept learning in neuromorphic vision systems: What can we learn from insects? *Journal of Software Engineering and Applications*, 7, 387–395. [http://dx.doi.org/10.4236/jsea.2014.75035](http://dx.doi.org/10.4236/jsea.2014.75035)

Schmickl, T., Bogdan, S., Correia, L., Kernbach, S., Mondada, F., Bodí, M., ... Thenius, R. (2013). ASSISI: Mixing animals with robots in a hybrid society. *Biomimetic and Biohybrid Systems Lecture Notes in Computer Science*, 8064, 441–443. [http://dx.doi.org/10.1007/978-3-642-39802-5](http://dx.doi.org/10.1007/978-3-642-39802-5)

Schmickl, T., Szoppek, M., & Bodí, M. (2013). ASSISI: Charged hot bees shakin’ in the spotlight. *Self-Adaptive and Self-Organizing Systems (ASAO)*, 2013, 259–260.

Sinclair, R. S. (2010). Thinking and writing: Cognitive science and intelligence analysis. Revised edition of a monograph originally, published by CSI in 1985. Washington, DC: Centre for the Study of Intelligence.

Sinclair, R. S. (2011). *Thinking and writing: Cognitive science and intelligence analysis (intelligence and counterintelligence studies)*. New York, NY: Nova Science.

Vanitha, A., & Niraimathi, S. (2013). Study on decision tree competent data classification. *International Journal of Computer Science and Mobile Computing (IJCSMC)*, 2, 365–370.