Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 3, 2022, 385-400.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Automatic Control and Guidance of Mobile Robot using Machine Learning Methods

Somaye Ghandi* and Hadi Mokhtari

Department of Industrial Engineering, Faculty of Engineering, University of Kashan, Kashan, Iran.

Article Info

Article History:
Received 13 October 2021
Revised 31 January 2022
Accepted 13 March 2022

DOI:10.22044/jadm.2022.11278.2286

Keywords:
Guidance of Mobile Robot, Classifier, Parametric Approach, Semiparametric Approach, Non-parametric Approach.

* *Corresponding author: s.ghandi@kashanu.ac.ir (S. Ghandi).*

Abstract

In many applications of the robotics, the mobile robot should be guided from a source to a specific destination. The automatic control and guidance of a mobile robot is a challenge in the context of robotics. Thus, in the current work, this problem is studied using various machine learning methods. Controlling a mobile robot is to help it to make the right decision about changing direction according to the information read by the sensors mounted around the waist of the robot. The machine learning methods are trained using 3 large datasets read by the sensors and obtained from the machine learning database of UCI. The methods employed include (i) discriminators: greedy hypercube classifier and support vector machines, (ii) parametric approaches: Naive Bayes' classifier with and without dimensionality reduction methods, (iii) semiparametric algorithms: expectation-maximization (EM) algorithm, *C*-means, *K*-means, agglomerative clustering, (iv) non-parametric approaches for defining the density function: histogram and kernel estimators, (v) non-parametric approaches for learning: *k*-nearest neighbors and decision tree, and (vi) combining multiple learners: boosting and bagging. These methods are compared based on various metrics. The computational results indicate the superior performance of the implemented methods compared to the previous ones using the mentioned dataset. In general, boosting, bagging, unpruned tree, and pruned tree ($\theta = 10^{-7}$) have given better results compared to the existing ones. Also, the efficiency of the implemented decision tree is better than the other employed methods, and this method improves the classification precision, TP-rate, FP-rate, and MSE of the classes by 0.1%, 0.1%, 0.001%, and 0.001%.

1. Introduction

Robotics plays an essential role in reducing the human efforts and increasing utility in various areas. In addition, due to developments in the area of mobile robots, the complexity of many problems is reduced significantly. As the examples, planet or submarine exploration [1], operation in urban areas [2], and unmanned flight [3] can be mentioned. In all of the above applications, the main purpose is to guide an object. Guidance is to determine the trajectory of an object from a starting point to a target point. Indeed, it should be mentioned that automating the guidance process is a challenging area in the context of robotics, and mobile robots are designed such that they can be employed in real complicated environments. In addition, a robot is only able to observe and understand its surrounding in a limited manner [4]. Therefore, designing an automatic guidance method for a mobile robot is one of the main challenges in this context. Many of the conventional techniques are not able to guide the mobile robots in the real-

world condition due to its complex nature. However, this problem can be solved using the machine learning techniques. Robot guidance through wall-following was introduced by Krishna and Kalra [5]. Bekey showed that the above concept could be formulated as a pattern recognition problem [6]. An automatic fuzzy controller using GA has been designed in [7] for mobile robot guidance through wall-following. The algorithm has been developed based on the Iterative Rule Learning (IRL) approach, in which the designer should determine the parameters like the accuracy of each variable and the objective function. However, there is no limitation in terms of the number of linguistic labels and values defined by the membership functions. Some studies including [8] have adjusted a set of fuzzy controller parameters including membership functions, ranking factors, and control laws for robot guidance using GA. In addition, Freire *et al*. [10] have proved that robot guidance through wall-following cannot be discriminated linearly. Then they have used a short-term memory mechanism in the static and dynamic neural networks in order to improve the classifiers employed for the mentioned problem [9]. In [10], a simple robot with local sensors that is able to move in a polygonal environment has been studied and analyzed. The robot moves in parallel with the sides of the polygon and moves inside the area. The main drawback of this approach is the lack of global sensors as a result of which, locating cannot be performed accurately. In addition, a Multi-Instance Multi-Label learning Gaussian Process (MIMLGP) algorithm has been presented in order to solve the automatic mobile robot guidance problem visually [11]. Chen *et al*. [13] have proposed a particle selection approach to search for a set of optimal parameters for designing the intelligent classifiers used to guide the robot through wall-following. The particle search method is able to achieve an accuracy higher than 90% compared to the conventional network search method [12]. In addition, Dash *et al*. [15] have proposed a controller based on Artificial Neural Network (ANN) for robot guidance that moves along the walls of a polygonal room using the information read by the sensors. The employed database is SCITOS G5, and the gradient descent strategy is used to learn the neural network [13]. Next, they have used Feed-Forward Neural Network (FFNN) based on the gravitational search to control the wall-following robot guidance [14]. In 2018, a control method was proposed for mobile robot guidance in an unknown environment that was based on reinforcement learning and dynamic group artificial bee colony comprising Behavior Manager (BM), Toward Goal (TG), and Wall-Following (WF) states [15]. BM is designed such that it is automatically changed to any of the other two states depending on the position of the mobile robot in the environment. When an obstacle is detected, the mobile robot goes to the WF state; otherwise, it moves towards the goal using the TG state. Although all of the above methods are able to guide the mobile robot using a small number of input data, when there is a large amount of data, the accuracy of these methods is reduced significantly.

Of course, it should be noted that the autonomous robot navigation of different types of robots with different applications and using different methods has been done in articles such as [16-22].

Machine learning is programming the computers to optimize a performance criterion using example data or past experience. Learning is required in the cases where a computer program cannot be directly written to solve a given problem but example data or experience is required. One case where learning is necessary is when the problem to be solved depends on the particular environment or changes in time. In these situations, having the general-purpose systems that can adapt to their circumstances is preferred to explicitly writing a different program for each special circumstance. In other words, machine learning is a part of artificial intelligence that enables the systems to have the ability to learn in a changing environment, and helps the system designer need not foresee and provide the solutions for all possible situations [23].

Already, there are many successful applications of machine learning in various domains. One of these domains is robot navigation, in which the robots learn to optimize their behavior to complete a task using the minimum resources. In this work, the wall-following mobile robot navigation is done by implementing and employing the machine learning approaches. The robot is guided using the ultrasonic sensors installed at the waist of the robot and through following the walls of a closed room. The main purpose of this work is to increase the robot guidance accuracy in an unknown environment.

The rest of this paper is organized as what follows. Section 2 describes the employed dataset briefly. Section 3 introduces the classifiers employed to improve the performance of the robot guidance through wall-following. Section 4 includes the computational results of applying different classifiers and their analysis. Section 5

concludes the paper, and presents some suggestions for the future works.

2. Employed Dataset

The dataset employed in this work is "wall-following robot navigation data", accessed through UCI machine leaning repository [24]. SCITOS G5 is used in order to collect the data. The robot includes a belt comprised of 24 ultrasonic sensors. The sensor in the front of abdomen of the robot is sensor #1, and number of the sensors increases clockwise. The robot rotates 4 cycles clockwise in the environment, and makes 9 observations per second. The observations of these sensors are the attributes of the dataset [9]. The robot is modeled as a simple kinematic unicycle with state $q = [x \quad y \quad \theta]$, where (x, y) is the Cartesian location of the robot's center within the world model and θ is the robot's yaw angle as measured from the wall or the world model's X -axis. Figure 1 shows the arrangement of the objects located in the test room of the employed dataset. In order to analyze the employed methods, 3 different datasets are used, which are as follow:

- Sensor readings 24-dataset including the values measured by all 24 ultrasonic sensors.
- Sensor readings 4-dataset including the values measured by 4 sensors installed at the back, front, right, and left.
- Sensor readings 2-dataset including the values measured by 2 sensors at the front and left of the robot.

In order to guide and label the behavior of following walls based on the sensory readings at a given time step, a navigation algorithm is used, which is responsible for generating the decisions that the robot has to take along its navigation. This algorithm makes use of heuristic If-Then rules based on the measures of the variables *front distance* and *left distance*. Each one of these distances is not calculated by a single sensor but by a number of them and the distance used is the smallest found by those sensors [9]. Based on the amounts of these variables, one of the following moves (or classes) are selected:

1- Class 1: If $0.55\ meter\ (m) \le left\ distance \le 0.9m$ and $front\ distance > 0.9m$, then move-forward.

2- Class 2: If the $left\ distance < 0.55m$ and the $front\ distance > 0.9m$, then slow down, and turn to the right (slight-right-turn).

3- Class 3: If the $front\ distance \le 0.9m$, then stop and turn to the right (sharp-right-turn).

4- Class 4: If the $left\ distance > 0.9m$ and the $front\ distance > 0.9m$, then slow down, and turn to the left (slight-left-turn).

Table 1 shows the distribution of the samples in each class of the dataset. All datasets have the same number of samples. As it can be seen in the table, the number of samples of the dataset is 5456, where 2205 of them belong to class 1, 826 of them belong to class 2, 2097 of them belong to class 3, and 328 of them belong to class 4.

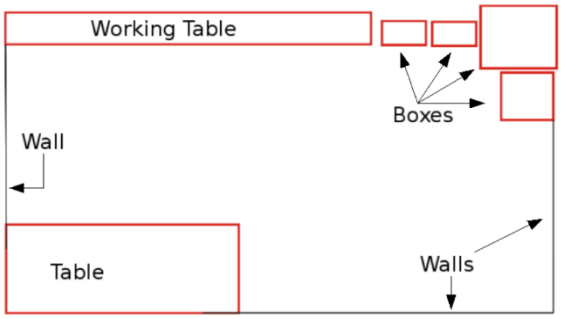


Figure 1. Sketch of robot's navigation environment (from [9]).

Table 1. Distribution of samples in each of the 4 classes of the used datasets.

Class level	Move-forward	Slight-right-turn	Sharp-right-turn	Slight-left-turn
No. of instances	2205	826	2097	328
Percentage	40.41	15.14	38.43	6.01

3. Using Classifier Algorithm to Improve Performance

As mentioned earlier, the purpose of employing the learning algorithms is to guide the robot based on the observations of the sensors. These algorithms are the supervised learning algorithms (in contrast to the unsupervised learning algorithms), in which there exist an input *x* and an

output y, and the algorithm learns the map, model or transfer function as $y = g(x \mid \varphi)$, in which $g(\cdot)$ is the model of interest and φ shows the parameters of the considered model. Supervised learning can be divided into the regression learning and divisive learning based on the output of the transfer function. In regression, y and $g(\cdot)$ are a number and a regression function, and in the divisive model, y and $g(\cdot)$ are the code of the corresponding class and the discriminator function of the samples of different classes. The machine learning algorithm aims to optimize the parameters of the model such that the estimation error is minimized. Minimizing the estimation error means making the predicted values (estimated values) as close as possible to the real values [23].

According to the above discussion, the algorithms used for robot guidance are the classifier algorithms, in which the input is the data obtained from the sensors and the output is the code of the corresponding class. In this work, 5 classifier algorithms are used in order to improve the robot guidance that are introduced in the following, and their results are represented. In all of these algorithms, the following assumptions are considered:

- 70% of the data is used as the training set and 30% is used as the test set. The model is learned using the data of the training set, and then the learned model is evaluated according to the data of the test set.
- The test and training sets are generated randomly.
- The test and training sets are the same for all algorithms.
- The sets are selected such that the ratio of the classes in the test and training sets remains almost the same.

The employed algorithms include the discriminator algorithms, parametric algorithms, semi-parametric algorithms, non-parametric algorithms, and a combination of the multiple learners that are described in the following.

3.1. Discriminator algorithms

The discriminator algorithms discriminate the data of the test set, and assign them to one class of the dataset through the learning relationships and the laws governing the training set. Using these algorithms and based on the relationships governing the previous observations and performance of the robots about each observation (moving along each class), knowing the new data obtained from the observations of the sensors, the performance and correct movement of the robot can be predicted correctly. This class of discriminant algorithms includes greedy hypercube classifier and support vector machines (SVMs).

Greedy Hypercube Classifier- in this algorithm, first, a null set is created as the set of best samples of each one of the 4 classes. Then one of the samples belonging to a specific class is selected randomly from the training set at each step. Next, the mentioned sample is temporarily added to the set of best samples of the mentioned class, and eliminated from the training set. Then the minimum and maximum values of the 2, 4 or 24 attributes (for sensor readings 2 dataset, sensor readings 4-dataset, and sensor readings 24-dataset) in the set of best samples of the mentioned class are calculated, and a hypercube is developed, where its edges are the boundaries of the areas created with minimum and maximum values of different attributes. This means that the number of edges of the hypercube for any of the 3 datasets is equal to the number of attributes of that dataset (2, 4 or 24 attributes). Then the new sample is permanently added to the set of best samples of the mentioned class if it would improve one or a combination of the following performance criteria for the considered class:

- Maximizing number of true positives (TPs)
- Minimizing number of false positives (FPs)
- Maximizing $TP - FP$
- Maximizing precision (Precision $= TP / (FP + TP) \times 100\%$)

If one or several measures of the above measures are not improved by adding the new sample to the set of best samples of the corresponding class, the new sample is eliminated from this set. This procedure is continued until all samples are checked. The values of the measures obtained from applying this classifier on 3 robot guidance datasets are represented in Table 2 for any of the 4 classes existing in the datasets. As it can be seen in this table, the greedy hypercube classifier is able to obtain the optimal value for the considered measures in 36 cases (values in the table that are shown in bold) out of the total 48 cases, and for the other cases, the values of the measures are very close to the optimal values.

In addition, the results obtained from applying the greedy hypercube classifier on two attributes of the sensor readings 2 dataset (e.g., front distance and left distance) are shown in Figure 2. As it can be seen in this figure, the classes can be discriminated completely by the created

hypercubes (rectangles). The red, green, blue, and yellow marks represent the samples of class 1 (move-forward), samples of class 2 (slight-right-turn), samples of class 3 (sharp-right-turn), and samples of class 4 (slight-left-turn). This verifies the high accuracy and efficiency of the greedy hypercube classifier in discriminating the samples of any of the 4 classes of the robot navigation dataset.

Table 2. Results of implementation greedy hypercube on three datasets based on the common performance metrics.

	Performance criterion															
	TP				**FP**				**TP – FP**				**Precision (%)**			
Dataset	**Class level**															
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Sensor readings 2	**2205**	**826**	**2097**	**328**	**0**	**0**	**0**	**0**	**2205**	**826**	**2097**	**328**	**100**	**100**	**100**	**100**
Sensor readings 4	**2205**	**826**	**2097**	**328**	**0**	**0**	**0**	**0**	**2205**	**826**	**2097**	**328**	**100**	**100**	**100**	**100**
Sensor readings 24	2054	691	1628	**328**	51	20	26	**0**	2003	671	1602	**328**	97.58	97.19	98.43	**100**

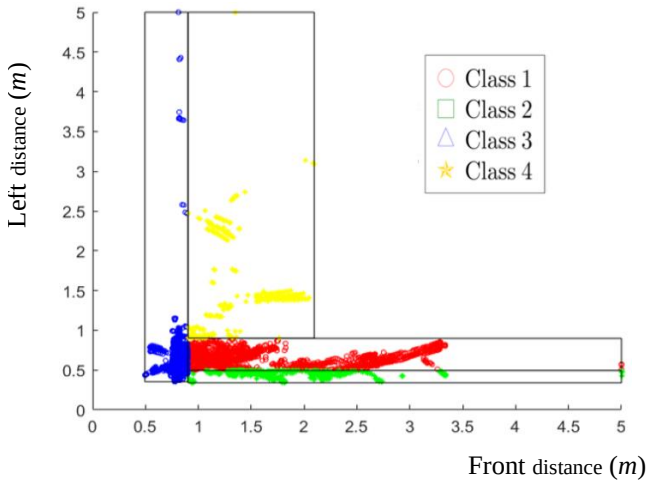


Figure 2. Results from applying greedy hypercube classifier on the sensor readings 2 dataset (attribute 1: front distance and attribute 2: left distance).

Support Vector Machines- this classifier is one of the kernel-based algorithms that has become popular in the recent years. Since these algorithms employ the kernel functions, they can be applied to a wide variety of applications, particularly in bioinformatics and language processing. SVM aims to find the subsets of the training set samples as the vectors that support the boundaries. This support vectors must be close enough to the boundaries that separate the adjacent classes. Due to this property, these vectors can be used to separate different classes properly. In this work, 3 kernel functions including normalized polynomial function (N-Poly), Puk kernel, and radial basis function (RBF) are used to create the support vectors. In addition, in order to compare the performance of different kernel functions, the following performance criteria are used:

- Maximizing ratio of TP to total samples $(TP-rate = TP/P)$
- Maximizing precision
- Maximizing TP-rate percentage to total samples $(Recall = TP/P * 100\%)$
- Minimizing number of support vectors (NSV = Number of Support Vectors)

The results obtained from employing the three mentioned kerl functions for the robot navigation database are given in Table 3. The cells regarding the minimization measures are colored blue. Also in this table, the best values of each performance criteria among all kernel functions are shown in white and shaded in green. The penalty factor (c) represents the penalty considered in the discrimination function for the samples that cannot be classified by the kernel function, and therefore, are incorrectly assigned to another class.

The results represented in the table indicate that for the employed datasets, N-Poly kernel function gives the best results for all performance criteria and penalty factors compared to the other kernel functions. It should be mentioned that by increasing the value of c from 1 to 1400, the values of all performance criteria are improved for all the three kernel functions. As it can be seen in the table, by increasing the penalty factor to 2000, the values of the measures either remain constant or worsen. Considering the above discussion and

the values of the table, N-Poly is the best kernel function among the three employed functions, and the best penalty factor is 1400 (values in bold).

Table 3. Results of implementing three kernel functions for support vector machines on the sensor readings 24 dataset.

Performance criterion	kernel function	Penalty factor (*c*)								
		1	10	20	22	30	100	1000	1400	2000
TP_rate	N-Poly	0.843	0.899	0.905	0.907	0.907	0.908	0.908	**0.912**	0.911
	Puk	0.743	0.768	0.774	0.744	0.774	0.838	0.848	0.879	0.879
	RBF	0.563	0.728	0.758	0.761	0.774	0.774	0.774	0.774	0.774
Precision (%)	N-Poly	76.7	77.8	83.8	89.8	90.3	90.1	90.2	**90.3**	90.2
	Puk	74.3	77.1	77.7	77.8	77.7	77.4	77.4	77.7	77.7
	RBF	64.3	73.4	76.3	76.7	77.8	83.8	89.8	90.3	90.1
Recall (%)	N-Poly	84.3	89.9	90.5	90.7	90.7	90.8	90.8	**91.2**	91.1
	Puk	74.3	76.8	77.4	77.4	77.4	83.8	84.8	87.9	87.9
	RBF	56.3	72.8	75.8	76.1	77.4	77.4	77.4	77.4	77.4
NSV	N-Poly	167	101	84	82	76	71	**66**	**66**	**66**
	Puk	367	231	203	197	168	124	85	79	76
	RBF	511	337	295	290	269	191	102	90	87

In addition to the before mentioned common metrics that are well-known and have been widely used in machine learning, the new suggested metrics other than those common metrics have been used to assess the performance of the algorithm. The goal of using these metrics is to evaluate the performance of the algorithm in other ways, and to obtain a balanced evaluation of the algorithm's performance [25]. These measures are:

- Youden's index (γ) that measures the ability of an algorithm to avoid failure as below:

$$\gamma = TP / P - (1 - TN / N) \quad (1)$$

in which TP, P, TN, and N are True Positives, total Positives, True Negatives, and total Negatives. A high value of γ indicates a better ability to avoid failure [26].

- Positive and negative likelihoods (LR_S) that are familiar epidemiologic measures and are useful and helpful for comparing two algorithms. Their advantage is that they evaluate the algorithm's performance with respect to both classes. The values of the positive likelihood (ρ^+) and negative likelihood (ρ^-) can be expressed as:

$$\rho^+ = \frac{TP / P}{(1 - TN / N)}, \ \rho^- = \frac{(1 - TP / P)}{TN / N} \quad (2)$$

A higher ρ^+ and a lower ρ^- indicate a better performance on the positive and negative classes, respectively [27].

- Diagnostic odds ratio (DOR) that is a global performance measure and is used in medicine for the comparison of diagnostic accuracies between two or more diagnostic tests [28]. It is calculated using the following equation:

$$DOR = \frac{TP * TN}{FP * FN} \quad (3)$$

A high value of DOR indicates a better ability to distinguish between the positive and negative examples.

The results of evaluating these performance metrics for the compared algorithms in this section are given in Table 4. These results indicate the better performance of the greedy hypercube classifier than the other compared algorithms.

3.2. Parametric algorithms

The main characteristic of the parametric algorithms is that they can be mathematically analyzed. In fact, these algorithms model the uncertainty between the samples and their corresponding class using probability rules. Accessing a training set, these algorithms can specify the distribution function of the mentioned dataset through determining a small number of parameters of the model (for example, mean and variance) using the maximum likelihood estimation. The determined distribution function is used to decide about the proper classification of the future samples (including the samples of the training set) [29]. In all algorithms of this section,

it is assumed that the samples of the training set follow the multi-variate normal distribution.

Table 4. Results of implementation of the greedy hypercube classifier and three kernel functions for support vector machines on the sensor readings 24 dataset based on γ , ρ^+ , ρ^- and Dor performance metrics.

Algorithm	Performance criterion			
	Youden's index (γ)	positive likelihood (ρ^+)	negative likelihood (ρ^-)	Diagnostic odds ratio (DOR)
Greedy hypercube classifier	**0.844**	**47.89**	0.141	**343.98**
N-Poly kernel function (c = 1400)	0.814	9.31	**0.098**	95.35
Puk kernel function (c = 1400)	0.627	3.49	0.162	21.55
RBF kernel function (c = 1400)	0.691	9.32	0.246	37.74

There are 4 possible cases for modelling the relationship between the classes for the multi-variate normal distribution:

1. All classes have a common diagonal covariance matrix (linear diagonal).
2. Each class has a separate diagonal covariance matrix (quadratic diagonal).
3. All classes have a common covariance matrix (linear).
4. Each class has a separate covariance matrix (quadratic).

The results obtained from modelling the data of the test set using the above cases are represented in Table 5 and Figure 3. The cells regarding the minimization measures are colored blue. It should be mentioned that in addition to the previous measures, the *FN* performance criterion is also used in this table that indicates the number of false negatives (number of samples that are incorrectly assigned to other classes), and is calculated as $FN = PTP$. In addition, since the test set includes 30% of the total samples (30% * 5456 = 1637), the number of samples of each class in this set would be 637, 238, 655, and 107, respectively. As it can be seen in the table, the fourth case (quadratic) is the best case for modelling the relationship between 4 classes of the dataset. This indicates that the covariance of the data existing in different classes is non-zero, and the variance of data distribution in different classes is also different. Also as it can be seen from Figure 3, the area under the precision-recall graph for Class 1 is more than the other classes. After that, the averaged over all classes and classes 3, 4, and 2, respectively, have the largest amount of area under the graph.

Table 5. Results obtained from modelling data of the test set using four cases for multi-variate normal distribution.

	Case	Diagonal linear					Diagonal quadratic					Linear					Quadratic				
	Class level	1	2	3	4	total	1	2	3	4	total	1	2	3	4	total	1	2	3	4	total
Criterion	TP	288	208	650	88	1234	528	227	605	101	1461	285	208	650	88	1231	529	227	605	101	**1462**
	FP	4	135	259	5	403	26	84	55	11	176	4	137	260	5	406	25	85	54	11	**175**
	FN	349	30	5	19	403	109	11	50	6	176	352	30	5	19	406	108	11	50	6	**175**
	Precision (%)	99	61	72	95	75.4	95	73	92	90	89.2	99	60	71	95	75.2	95	73	92	90	**89.3**
	Recall (%)	45	87	99	82	75.4	83	95	92	94	89.2	45	87	99	82	75.2	83	95	92	94	**89.3**

In the following, the results of implementing the parametric algorithm of the Bayesian networks without attribute reduction (including Naïve Bayes, Bayesian networks using covariance matrix, Bayesian networks using TAN, Tabu Search, and Hill Climber algorithms), and Bayesian networks with attribute reduction (including Linear Discriminant Analysis (LDA) and Principal Components Analysis (PCA)) are presented.

Bayesian Networks without Attribute Reduction Methods- in this section, various algorithms and classifiers are used in order to create the Bayesian networks, and the results are presented. The first classifier is the Naïve Bayes that ignores the correlation among the inputs and converts the multi-variate normal distribution model to a group of independent univariate normal distribution for all of the classes. Since, for the datasets used in this work, data independency is not correct according to the results of Table 5, this method gives worse results compared to the other algorithms used in this section. In the second classifier of this section, a Bayesian network is developed for the robot navigation dataset manually considering the relationships among the attributes and covariance matrix; the obtained network is compared with the Bayesian networks

resulting from the Tabu, TAN, and Hill Climber algorithms.

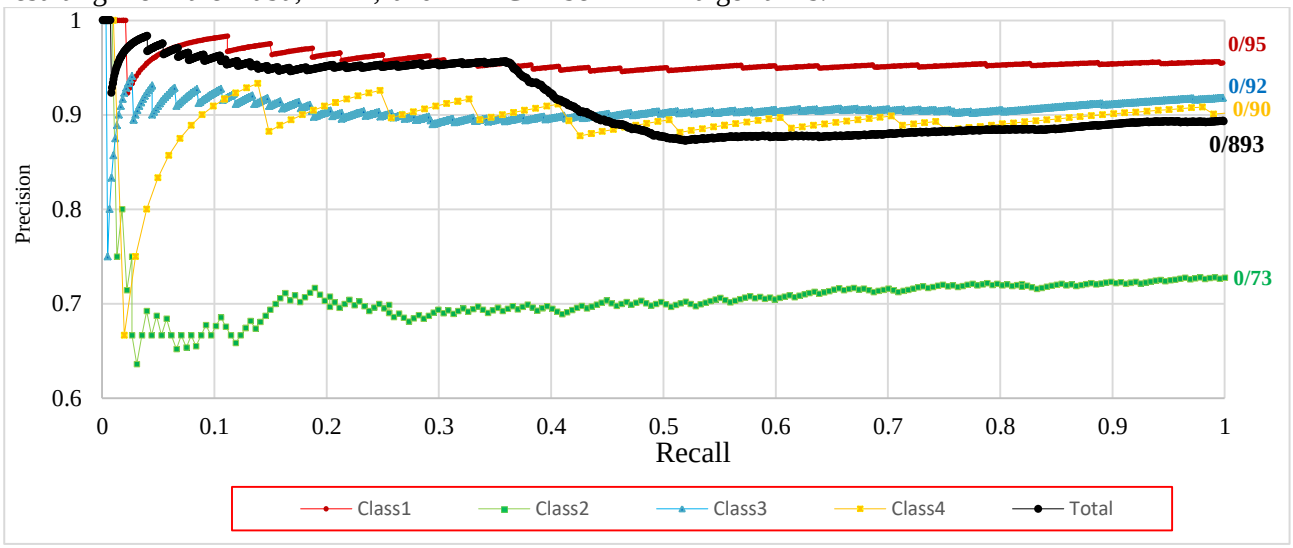


Figure 3. Precision-recall graph of four classes and averaged over all classes of the test set using the quadratic case.

It should be mentioned that one of the performance criteria used in this section is the Mean Squared Error (MSE) that is determined as $\frac{\sum_{i=1}^{m}(a_i - e_i)^2}{m}$, in which, a_i *and* e_i represent the real value and the estimated value of the sample i, and m is the number of estimated samples. The values of different measures for the 5 mentioned classifiers are represented in Table 6. As it can be seen in this table, TAN is the best method for creating a Bayesian network for the robot navigation dataset.

Table 6. Values of performance criteria for 5 Bayesian networks without attribute reduction.

Performance criterion / Bayesian network	TP-rate (%)	FP-rate (%)	Precision (%)	MSE
Naïve Bayes	53.6	15.4	77.7	0.13
Covariance matrix	82.7	9.0	90.2	0.06
TAN	**92.8**	**3.5**	**96.4**	**0.03**
Tabu Search	88.7	4.4	95.3	0.05
Hill Climber	88.9	4.4	95.3	0.05

Bayesian Networks with Attribute Reduction Methods- the time and computational complexity of each classifier depend on the number of inputs of that classifier. Therefore, one of the approaches to reduce the memory and computational complexity of a classifier is to reduce the dimension of the inputs of the problem (in this work, attribute reduction was used). There are two types of methods to reduce the number of attributes. The first type includes the attribute selection methods that select a subset of important attributes and eliminate other attributes. The second type includes the attribute extraction methods that generate smaller number of new attributes (k attributes) through combining the initial attributes (d attributes). The best and most applicable attribute extraction methods are the principal components analysis (PCA) and linear discriminant analysis (LDA) that are both used in this work for attribute reduction. In both of these methods, the primary space is linearly mapped to the new space. The main difference of these two methods is that the first method is unsupervised but the second one is one of the supervised learning methods. PCA is unsupervised since it specifies the discriminator function of the samples of different classes without considering class code of the samples.

In PCA, first, the class label of the samples is omitted, and then the dimensions of the attributes are reduced considering 80%, 90%, and 95% for the Proportion of Variance (POV) explained with k new attributes. Finally, learning is performed using the Bayesian network obtained from the TAN algorithm according to the new attributes. The results obtained from applying the PCA and LDA methods on the sensor readings 24 datasets are shown in Table 7. As mentioned earlier, the

number of initial attributes of this dataset equals to 24. The following six performance criterion are used for comparisons:

1. Precision,
2. Absolute Mean Error (AME) $\frac{\sum_{i=1}^{m}|a_i - e_i|}{m}$,
3. Mean Squared Error (MSE) $\frac{\sum_{i=1}^{m}(a_i - e_i)^2}{m}$,
4. Relative Absolute Error (RAE) $\frac{\sum_{i=1}^{m}|a_i - e_i|}{\sum_{i=1}^{m}|a_i - \bar{a}|}$,
5. Root Relative Squared Error (RRSE) $\sqrt{\frac{\sum_{i=1}^{m}(a_i - e_i)^2}{\sum_{i=1}^{m}(a_i - \bar{a})^2}}$,
6. Number of extracted attributes.

The value of $\bar{a}$ for the 4th and 5th criteria indicate the average of *m* estimated sample. As it can be seen in Table 7, the higher is POV, the amount of the precision and the number of extracted attributes increase, and the error values decrease. Unlike PCA, LDA does not specify the optimal number of attributes; thus the error and precision measures are calculated after reduction of 1 to 24 attributes. Using the obtained values, it can be concluded that the best case for this method is when the number of attributes is 19. In addition, as it can be seen in the table, the Bayesian network TAN method alone shows better results than when combining it with the attribute extraction PCA and LDA methods. The reason is that since the number of attributes in the considered dataset in this work is small in itself (equal to 24 attributes for sensor readings 24 dataset), the LDA and PCA methods have not been very effective on this dataset. Another important conclusion is that LDA gives better results compared to the equivalent (with equal number of attributes) PCA. The reason of this excellence is that LDA, unlike PCA, considers information of the samples' class to extract new attributes.

Table 7. Results of implementing TAN algorithm after attribute extraction methods.

Approach		PCA (pov = 0.8) + TAN	PCA (pov = 0.9) + TAN	PCA (pov = 0.95) + TAN	LDA + TAN
performance criterion	**1**	73.7%	74.4%	75.2%	**75.5%**
	2	0.155	0.146	0.143	**0.137**
	3	0.096	0.095	0.095	**0.091**
	4	47%	44%	43%	**42%**
	5	76%	76%	76%	**74%**
	6	**14**	19	22	19

3.3. Semiparametric algorithms

In the parametric algorithms in the previous section, it was assumed that the different samples followed a specific distribution function (multivariate normal distribution). When such condition is not established, the semi-parametric algorithms are used that mainly assume that the samples of a class are a combination of the known distributions, i.e. each class includes several groups, and each group follows a known parametric model. The clustering methods are used to determine the distribution parameters of different groups. In all methods of this section, the class to cluster matrix is formed first, where the number of its rows is equal to the number of classes, and the number of its columns is equal to the number of clusters. In order to form this matrix, the probability that each sample belongs to different clusters is checked, and the sample is considered to belong to the cluster with the maximum probability. This process is repeated for all samples of a specific class. Then the value of the i^{th} row and the j^{th} column of the matrix represents the total number of samples of class i considered to belong to cluster j. Considering the information of this matrix, the classes are assigned to the clusters. In order to assign the classes to the clusters, the maximum value of the class to cluster matrix that has not been assigned yet is selected, and its corresponding class is assigned to the corresponding cluster. Then the row of the maximum value is eliminated. This process continues until all classes are assigned to the proper cluster.

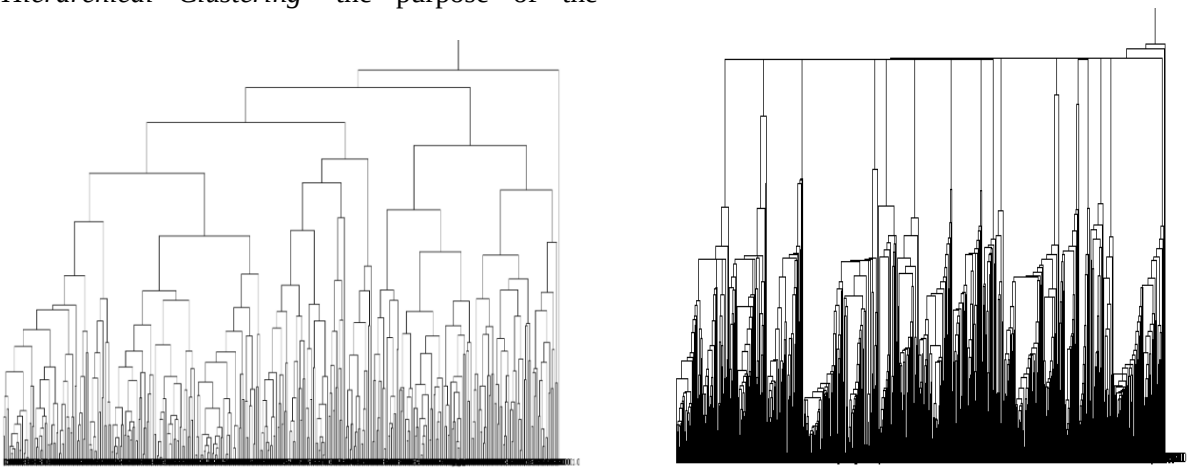
Unlike the attribute extraction methods introduced in the previous section that found the correlation among different attributes and classified them, the clustering methods find the similarity of the samples and classify them. The similarity of these methods is that both methods are performed as an unsupervised pre-processing step before the learning step, and aim to determine a map from the primary space to a new space (with a smaller dimension compared to the primary space). The

advantage of using the unsupervised methods is that it is not required to label the data (that is time-consuming), i.e., using this method, a large number of unlabeled data is used to determine the map, and then the main step (classification of the samples of different classes) is performed on a smaller number of labeled data using a supervised learner (a Bayesian network using the TAN algorithm). It should be noted that the dataset used in this section is sensor reading 24, and therefore, the number of attributes equals to 24. In the following, each clustering algorithm is discussed briefly:

C-mean and K-means- the purpose of these algorithms is to determine the *C* or *K* basis vector that minimize the recovery error. In order to determine *C* or *K*, the values smaller than 4 (the number of classes) are investigated. The results indicate that the best number of basis vectors equals to 4. This means that each one of 4 classes is assigned to a different cluster.

Expectation-Maximization algorithm (EM)- the purpose of this algorithm is to find the parameters of the model such that likelihood is maximized.

Hierarchical Clustering- the purpose of the hierarchical clustering methods is to find the groups in which its samples are more similar compared to the samples of the other groups. These methods might use either agglomerative clustering or divisive clustering. A divisive clustering algorithm starts with only one group including all the training samples, and divides larger groups to smaller groups until *N* groups are achieved. On the other hand, an agglomerative clustering algorithm starts with *N* groups, where each one includes one the training sample and integrates adjacent groups until only one group is achieved. In the single-link clustering, the distance between two groups is the minimum distance between all pairs of samples of the two groups, and in the complete-link clustering, the distance between two groups is the maximum distance between all pairs of samples in the two groups. The output of an agglomerative algorithm is represented as a hierarchical structure called dendogram, which is a tree representation in which the leaves indicate the samples and are grouped in order of integration. Figure 4 shows the dendogram of divisive single-link and divisive complete-link clustering for the robot navigation dataset.



(a) (b)

Figure 4. Dendrogram derived from implementation of a divisive single-link clustering method (a) and a divisive complete-link clustering method (b) for robot navigation dataset.

Table 8. Results of implementation of different clustering algorithms on robot navigation dataset.

Algorithm	*C*-means	Agglomerative (complete-link)	EM	Agglomerative (single-link)	*K*-means
Percentage of incorrectly classified samples	**49.7%**	53.4%	59.3%	59.4%	60.4%

The considered performance criterion minimizes the percentage of the samples that are classified incorrectly after clustering, and the classification and the results are given in Table 8. The results indicate that assuming groups at the input data and using the clustering methods for exploring these groups is not a suitable assumption. The reason is that the belonging samples to different classes are tangled, and also the dispersion of samples of a class about their mean is high.

3.4. Non-parametric algorithms

In the algorithms discussed in the previous sections (parametric algorithms and semi-parametric algorithms), it was assumed that

different samples followed one or a combination of functions with known distribution. When this condition is not satisfied, non-parametric algorithms are used that estimate the density function of the samples before classifying the data. The main assumption of these algorithms is that similar inputs generate similar outputs. Therefore, in order to determine the class of a sample in the test set, similar samples of the training set are determined based on the proper distance measurement metric, and then the class of the considered sample is determined through interpolating classes of similar samples in the training set. The difference of various non-parametric algorithms is in determining similar samples of the training set and interpolating the output value (class) of the samples to determine the class of the current sample. In a parametric algorithm, it is assumed that all the training samples affect the final global estimation. However, in the non-parametric algorithms, there is not a unique global model but local models are determined when required, and they are only affected by training the data in their neighborhood (based on the distance measurement metric). It is obvious that these algorithms are among the supervised learning methods. In the following, the estimating density function of sample distribution using various methods is discussed, and then the results of non-parametric *K*-Nearest Neighbor (KNN) and decision-tree algorithms based on these density functions for classifying the samples of the test set are presented.

Estimating Density Function of Sample Distribution- in this section, assuming that the samples are selected from some unknown distributions, independently, the density function of these distributions is estimated using the histogram estimator and kernel estimator. In the histogram method, the space is divided into a number of intervals of equal size, and the abundance of each interval is equal to the number of samples in it. In the kernel estimator, in order to obtain a smooth estimation, a uniform weighting function called kernel function is used, where the most well-known such function is the Gaussian kernel. The parameter h describes the length of the effectiveness interval of the samples. Figure 5 shows the results of applying the two introduced estimators for determining the density function distribution of the 1th attribute of the sensor readings 24 dataset for different numbers of intervals (50, 500) and different values of parameter h (0.05, 0.12, 1) for the Gaussian kernel. As it can be seen, as the value of h increases, the overlap and effectiveness of different samples increase and the estimated density function becomes smoother. As it can be seen from this figure, the front distance of the robot to the walls, which is the 1th attribute of the sensor readings 24 dataset, has a normal distribution with an average of 1.5. This result is completely consistent with the intervals of this attribute values, as is shown in Figure 2. Also it is necessary to mention that this attribute is a continuous variable similar to the other 23 features of the considered dataset.

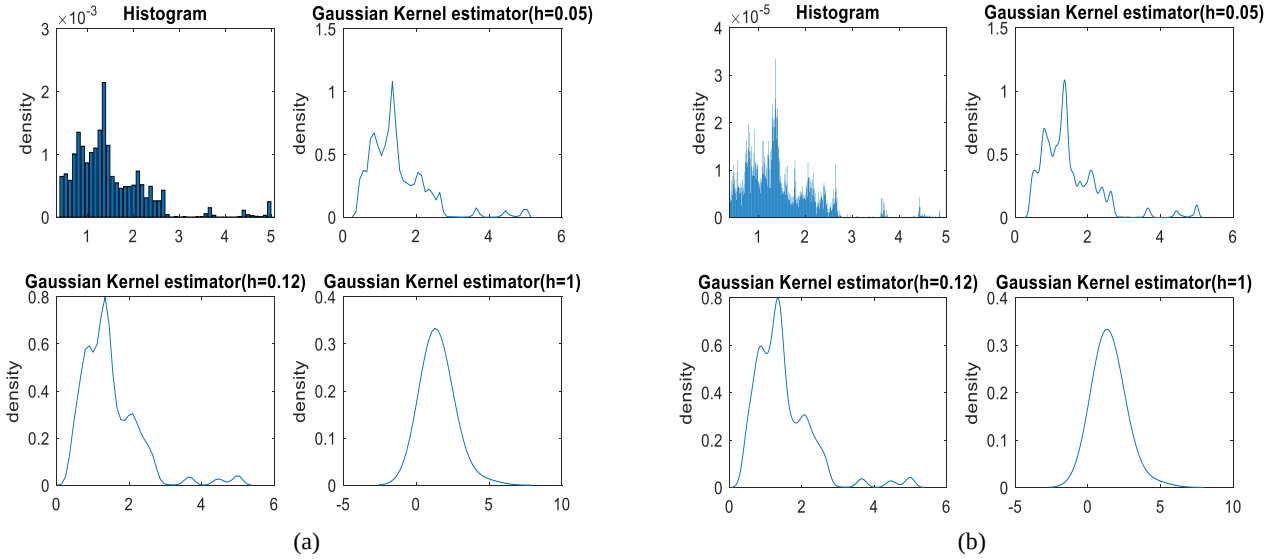


Figure 5. Results of running histogram and kernel estimators for 1th attribute of the sensor readings 24 dataset with (a) 50 intervals and (b) 500 intervals.

Learning through Nonparametric Algorithms- in this section, the results of employing non-parametric algorithms of *K*-Nearest Neighbor (KNN) and decision-tree for discriminating the samples of the test set are represented. The KNN classifier assigns a sample to a class where the maximum number of samples among *K* neighbors of the considered sample belong to that class. The mentioned algorithm is executed for K = 1, 3, 5, 7, 11, and the results are given in Table 9. It can be seen that for K = 1, the algorithm has the highest precision and recall and the minimum MSE because, as it can be seen in Figure 2, two neighboring samples with a very small distance might belong to different classes. Therefore, if a larger number of neighbors of a sample are used to determine the class of the considered sample, the error is increased, and the precision and recall are reduced.

Table 9. Implementation results of KNN algorithm for discriminating the robot control data.

K = 1				K = 3				K = 5				K = 7				K = 11			
Performance criteria*																			
1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
86.5	**6.9**	**92.6**	**0.047**	85.9	7.8	91.7	0.056	83.6	8.4	90.9	0.058	82.2	9.5	89.6	0.059	81.7	9.7	89.4	0.063

*Performance criteria: 1- TP-rate (%), 2- FP-rate (%), 3- Precision (%), 4- MSE

The decision-tree algorithm is a hierarchical data structure that employs the divide and conquer strategy, and creates a tree using the data of the training set, which can be converted to a set of simple and understandable laws. A decision tree is comprised of the middle decision nodes and the final nodes (leaves). At each decision node, a test function with discrete output is executed, and the branches are labeled. For a sample, existing in the test set as the input of the decision tree, one branch is selected based on the output value. This process starts from the root node, and continues until a leaf node is reached. Then the value written in the leaf node is reported as the input sample class. In a univariate tree, in the test function of each middle node, only one of the attributes (numerical attributes, in this work) is considered. For a specific dataset, there might be a large number of trees that can code the samples without error. The purpose is to find the minimum size tree. The measures used to determine the size of the tree are the number of middle nodes and the number of leaves. One of the operations that reduces this measure is pruning, which can perform as pre-pruning or post-pruning. In the first method, the tree construction is stopped before the tree is completed, and in the second method (that is usually better than the first one), the tree is constructed completely, and then the unnecessary sub-trees are determined and pruned.
In this work, the C4.5 univariate tree construction algorithm is used, which is a greedy method and selects the best branch at each step. In addition, the method used to reduce the size of the tree is the post-pruning method. The measure that shows the amount of pruning the tree is the reliability parameter (θ), which might be in the range of [0, 1]. For instance, θ = 0.05 for the pre-pruning method indicates when number of the remained training samples at a specific node is less than or equal to 5% of the training set, no branch originates from this node, and the mentioned node is considered as a leaf node. The above algorithm is executed for $\theta = 10^{-7}$, 10^{-6}, 1, and the results are given in Table 10. It is clear that as this parameter decreases, the size of the tree (total number of nodes) and the number of leaves is reduced but other criteria are worsened. In addition, it can be seen that the performance criteria for θ = 1 and $\theta = 10^{-6}$ are equal.

Table 10. Implementation results of decision-tree algorithm for discriminating the robot navigation data.

θ = 1						$\theta = 10^{-6}$						$\theta = 10^{-7}$					
Performance criteria*						Performance criteria*						Performance criteria*					
1	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**3**	**4**	**5**	**6**
99.57	**0.001**	**99.57**	**0.003**	35	18	**99.57**	**0.001**	**99.57**	**0.003**	35	18	99.27	0.003	99.27	0.005	**31**	**16**

* 1- TP-rate (%), 2- FP-rate (%), 3- Precision (%), 4- MAE, 5- Number of Nodes, 6- Number of Leaves.

The selected attributes from the sensor readings 24 dataset for the unpruned tree include the attributes 9, 10, 11, 12, 13, 14, 15, 18, 19, and 20, and other attributes are not used for constructing the tree. For the pruned tree, attribute No. 13 is eliminated from the above list, and it is added to the list of attributes that were not used during the tree construction.

3.5. Combining multiple learners

Each learning algorithm considers a specific model with a set of assumptions that might result in error if the assumptions do not match with the data. In addition, there are some cases for which even the best learners are not accurate enough. Therefore, by suitable combining several basic learners, the accuracy can be increased. In this section, the models comprising several learners are introduced; these learners complement each other such that their combination gives better results (based on the performance criteria). There are two types of methods for combining several learners:

1. Multi-expert combination methods have basic learners that operate in parallel. These methods are either global or local. In the global methods known as the learner fusion methods (for example, voting methods and stacking methods), all the basic learners generate an independent output for each input sample, and all these outputs are used to generate the final output. In the local methods that are also known as learner selection (for example, mixture of experts), there is a gate model that selects one or a few numbers of the basic learners to generate the final output for one input sample.
2. Multi-stage combination methods employ a series method in which the next basic learner is trained using the samples for which the previous learner was not accurate enough. In these methods (for example, cascading method), the basic learners are sorted in an ascending order based on their complexity, and a more complicated basic learner is used if the previous simple learners are not reliable enough.

In this work, the bagging and boosting methods are used to combine the learners. The bagging method is a voting one in which a different training subset is considered for each basic learner. The samples of each subset are determined among the samples of the training set through random selection with paste. The samples of each subset are similar to each other because they are selected from a main set. On the other hand, these samples are a bit different due to the random selection. In the boosting method, which is also a voting method, the next learner is trained using the mistakes of the previous learners to generate the complementary basic learners. In both methods, the basic learners of Naïve Bayes, SVM, KNN, pruned, and unpruned decision trees are used, and the results are shown in Table 11. It should be mentioned that the values of parameters of each basic learner is considered to be equal to an optimal value mentioned in the previous sections. Both methods have shown better results compared to the basic learners alone. In addition, it should be mentioned that the boosting method has a higher precision and recall, and a smaller error compared to the bagging method.

Table 11. Results of combining Naïve Bayes, SVM, pruned, and unpruned decision tree using the bagging and boosting methods for classifying the robot navigation data.

Combination method	Bagging				Boosting			
Performance criterion	TP-rate	FP-rate	Precision	RAE	TP-rate	FP-rate	Precision	RAE
	99.6%	0.001%	99.6%	1.23%	**99.9%**	**0%**	**99.9%**	**0.18%**

4. Computational Results

All of the algorithms mentioned in Section 3 are applied to the wall-following robot navigation dataset, and the results of the best algorithm are given in Table 12. The algorithms are coded in MATLAB and Weka, and run on an Intel™ Core-i7 1.8 GHz CPU with 8 GB of RAM. The performance of these algorithms was compared to that of various algorithms existing in the literature, including:

1. LP(2, 4) (without STM): that indicates the perceptron neural network without short-term memory, with 2 input neurons and 4 output neurons [9].
2. ME(2, 4, 4) (without STM): that indicates a hybrid network of 4 experts without short-term memory, with 2 input neurons and 4 output neurons [9].
3. MLP(2, 6, 4) (without STM): that indicates a multi-layer perceptron neural network with 6 hidden neurons, without short-term memory, with 2 input neurons, and 4 output neurons [9].
4. LP(20, 4) (with STM): that is a perceptron neural network with short-term memory, 20 input neurons, and 4 output neurons [9].

5. Elman (2 + 4, 4, 4) (with STM): that represents an element network with 6 inputs and 4 output neurons.
6. Gradient Boost Classifer (GBC) by [30-31].

The following points can be concluded from Table 12.

- The table is ranked based on the precision of the learners.
- The improvement regarding the best reported results based on the precision, TP-rate, FP- rate, and MSE are 0.1%, 0.1%, 0.001%, and 0.001%. In other words, the precision and TP-rate are improved from 99.8% to 99.9%, FP-rate is improved from 0.001% to 0%, and MSE is improved from 0.002% to 0.001%.
- Boosting, Gradient Boost Classifer (GBC), bagging, unpruned Tree, and pruned tree ($\theta = 10^{-7}$) have given better results compared to the existing results (grey rows). This ranking indicates that the employed decision-tree properly models the discriminant boundaries for the robot navigation dataset.
- The performance of the applied boosting method in this work is better than the similar Gradient Boost Classifer (GBC) method in [30]. This is due to the tuning the parameters of each basic learner (e.g., Naïve Bayes, SVM, KNN, pruned and unpruned decision trees), and using these improved values of parameters in the introduced boosting method.

Table 12. Implementation results of different algorithms on the sensor reading 24 dataset.

Reference	method	Performance criteria				Rank
		TP-rate (%)	FP-rate (%)	Precision (%)	MSE	
[9]	**LP(2, 4) (without STM)**	42.7	55.0	43.7	0.66	13
	ME(2, 4, 4) (without STM)	5.22	74.7	0.07	0.75	14
	MLP(2, 6, 4) (without STM)	97.6	1.8	98.2	0.03	6
	LP(20, 4) (with STM)	67.1	41.2	61.2	0.23	12
	Elman(2 + 4, 4, 4) (with STM)	96.2	1.9	98.1	0.06	7
[30]	**Gradient Boost Classifier (GBC)**	**99.8**	**0.001**	**99.8**	**0.002**	**2**
This article	**Greedy Hypercube**	86.2	10.8	88.9	0.11	11
	SVM (C = 1400)	91.2	9.8	90.3	0.09	10
	Bayesian networks (TAN)	92.8	3.5	96.4	0.03	8
	KNN (k = 1)	86.5	6.9	92.6	0.05	9
	Unpruned tree	**99.6**	**0.001**	**99.6**	**0.003**	**4**
	Pruned tree ($\theta = 10^{-7}$)	**99.3**	**0.003**	**99.3**	**0.005**	**5**
	Bagging	**99.6**	**0.001**	**99.6**	**0.002**	**3**
	Boosting	**99.9**	**0**	**99.9**	**0.001**	**1**

4. Conclusions

In this work, the adaptive navigation of the mobile robot was investigated using various machine learning methods. Various algorithms were trained using the information of 3 datasets with 2, 4, and 24 attributes for the "wall-following navigation task with the mobile robot SCITOS-G5" dataset, and predicted the future control strategy for the SCITOS robot. The robot navigation could be modelled as a pattern recognition problem in which the models were the observations of the sensors, and the classes were the operations performed by the robot. The employed methods included the following algorithms:

- Greedy hypercube classifier and support vector machine classifier with different penalty factors
- Bayesian network parametric algorithms (including Naïve Bayes, covariance matric, TAN, Tabu search, hill climber) with and without the attribute extraction methods (including PCA and LDA)
- Semi-parametric algorithms including EM, *C*-means, *K*-means, agglomerative (single and complete link)
- Non-parametric algorithms used to determine the density function (including histogram and kernel estimators) and non-parametric learning (including KNN with different values of K, pruned and unpruned decision-tree).

- Voting methods for combining several learning algorithms including the bagging and boosting efficiency of the above methods were compared based on various efficiency measures (including TP-rate, FP-rate, precision, recall, MSE, and …). In general, the efficiency of the decision-tree is better than the other methods employed in this work and the previous works, and this method improves the classification precision, TP-rate, FP-rate, and MSE of the classes by 0.1%, 0.1%, 0.001%, and 0.001%. Using the bagging and boosting methods improves the efficiency of the classifiers; the boosting method improves the efficiency more than the bagging method. Among the attribute extraction methods used for the robot navigation dataset, LDA performs better than PCA.

In the future works, other machine learning algorithms can be used to improve the performance criteria, and achieve an accuracy of 100%. In addition, in order to combine the learners, other combination algorithms like stacking, mixture of experts or cascading can be used. The last suggestion for improvement is using other attribute reduction methods such as subset selection and attribute extraction methods such as Factor Analysis (FA), Multi-Dimensional Scaling (MDS), isometric attribute mapping (Isomap), and Locally Linear Embedding (LLE).

References

[1] D. M. Helmick, S. I. Roumeliotis, Y. Cheng, D. S. Clouse, M. Bajracharya, and L. H. Matthies, "Slip-compensated path following for planetary exploration rovers," *Advanced Robotics*, vol. 20, no. 11, pp. 1257-1280, Jan 2006.

[2] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Autonomous robots*, vol. 18, no. 1, pp. 81-102, Jan 2005.

[3] P. Fabiani, V. Fuertes, A. Piquereau, R. Mampey, and F. Teichteil-Königsbuch, "Autonomous flight and navigation of VTOL UAVs: from autonomy demonstrations to out-of-sight flights," *Aerospace Science and Technology*, vol. 11, no. 2-3, pp. 183-193, March 2007.

[4] M. Knudson, and K. Tumer, "Adaptive navigation for autonomous robots," *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 410-420, Jun 2011.

[5] K. M. Krishna, and P. K. Kalra, "Spatial understanding and temporal correlation for a mobile robot," *Spatial Cognition and Computation*, vol. 2, no. 3, pp. 219-259, Sep 2000.

[6] G.A. Bekey, "Autonomous robots: from biological inspiration to implementation and control," *MIT press*, May 2005.

[7] M. D. Mucientes, L. Moreno, A. Bugarín, and S. Barro, "Design of a fuzzy controller in mobile robotics using genetic algorithms," *Applied Soft Computing*, vol. 7, no. 2, pp. 540-546, March 2007.

[8] S. F. Desouky, and H. M. Schwartz, "Genetic-based fuzzy logic controller for a wall-following mobile robot," in *2009 American Control Conference*. Jun 2009, pp. 3555-3560.

[9] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study,". in *2009 6th Latin American Robotics Symposium (LARS 2009)*. Oct 2009, pp. 1-6.

[10] A. Katsev, B. Yershova, R. Tovar, Ghrist, and S. M. LaValle, "Mapping and pursuit-evasion strategies M. for a simple wall-following robot," *IEEE Transactions on robotics*, vol. 27, no 1, pp. 113-128, Jan 2011.

[11] J. He, H. Gu, and Z. Wang, "Multi-instance multi-label learning based on Gaussian process with application to visual mobile robot navigation," *Information Sciences*, vol. 190, pp. 162-177, May 2012.

[12] Y.-L. Chen, J., Cheng, C., Lin, X.,Wu, Y., Ou, & Y. Xu, "Classification-based learning by particle swarm optimization for wall-following robot navigation," *Neuro-computing*, vol. 113, pp. 27-35, Aug 2013.

[13] T. Dash, S. R., Sahu, T., Nayak, and G. Mishra, "Neural network approach to control wall-following robot navigation," in 2014 *IEEE International Conference on Advanced Communications, Control, and Computing Technologies*. May 2014, pp. 1072-1076.

[14] T. Dash, T. Nayak, and R.R. Swain. "Controlling wall following robot navigation based on gravitational search and feed forward neural network," in *Proceedings of the 2nd international conference on perception and machine intelligence*. Feb 2015, pp. 196-200.

[15] T.-C. Lin, C. C. Chen, and C. J. Lin, "Wall-following and navigation control of mobile robot using reinforcement learning based on dynamic group artificial bee colony," *Journal of Intelligent and Robotic Systems*, vol. 92, no. 2, pp. 343-357, Oct 2018.

[16] S. M. J. Jalali, S. Ahmadian, A., Khosravi, S. Mirjalili, M. R. Mahmoudi, and S. Nahavandi, "Neuroevolution-based autonomous robot navigation: a comparative study," *Cognitive Systems Research*, vol. 62, pp. 35-43, Aug 2020.

[17] N. Islam, K. Haseeb, A. Almogren, I. U. Din, M. Guizani, & A. Altameem, "A framework for

topological based map building: A solution to autonomous robot navigation in smart cities," *Future Generation Computer Systems*, vol. 111, pp. 644-653, Oct 2020.

[18] S. M. J. Jalali, R., Hedjam, A. Khosravi, A. A. Heidari, S. Mirjalili, and S. Nahavandi, "Autonomous robot navigation using moth-flame-based neuroevolution,*"* in *Evolutionary Machine Learning Techniques*. Springer. pp. 67-83, 2020.

[19] M. L. Lagunes, O., Castillo, J., Soria, and F. Valdez, "Optimization of a fuzzy controller for autonomous robot navigation using a new competitive multi-metaheuristic model," *Soft Computing*, pp. 1-20, March 2021.

[20] S. Tiwari, Y. Zheng, M. Pattinson, M. Campo-Cossio, R. Arnau, D. Obregon,... and J. Reyes, "Approach for Autonomous Robot Navigation in Greenhouse Environment for Integrated Pest Management,*"* in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. Apr 2020, pp. 1286-1294.

[21] S. M. J. Jalali, A. Khosravi, P. M. Kebria, R. Hedjam, & S. Nahavandi, "Autonomous robot navigation system using the evolutionary multi-verse optimizer algorithm,*"* in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. Oct 2019, pp. 1869-1875.

[22] S. M. J. Jalali, P. M., Kebria, A., Khosravi, K., Saleh, D., Nahavandi, and S. Nahavandi, "Optimal autonomous driving through deep imitation learning and neuroevolution,*"* in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. Oct 2019, pp. 1215-1220.

[23] E. Alpaydin, "Introduction to machine learning," *MIT press*, March 2020.

[24] A. Frank, and A. Asuncion, "UCI machine learning repository," URL http://archive.ics.uci.edu/ml.15, p. 22, 2010.

[25] A.B. Musa, "Comparative study on classification performance between support vector machine and logistic regression," *International Journal of Machine Learning and Cybernetics*, vol. 4, no**.** 1, pp. 13-24. Feb 2013

[26] M. Sokolova, N. Japkowicz, and S. Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Dec 2006, pp. 1015-1021.

[27] B. de Bragança Pereira, and C.A. de Bragança Pereira, "A likelihood aproach to diagnostic tests in clinical medicine," *REVSTAT–Statistical Journal*, vol. 3, no. 1, pp. 77-98. Jun 2005

[28] A. S. Glas, J. G. Lijmer, M. H., Prins, G. J. Bonsel, and P. M. Bossuyt, "The diagnostic odds ratio: a single indicator of test performance," *Journal of clinical epidemiology*, vol. 56, no. 11, pp. 1129-1135. Nov 2003

[29] P.E. Hart, D.G. Stork, and R.O. Duda, *"*Pattern classification,*" Wiley Hoboken*, 2000.

[30] I. Hammad, K. El-Sankary, and J. Gu."A comparative study on machine learning algorithms for the control of a wall following robot," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Dec 2019, pp. 2995-3000.

[31] M. Moradizirkohi, S. Izadpanah, (2017). "Direct adaptive fuzzy control of flexible-joint robots including actuator dynamics using particle swarm optimization," *Journal of AI and Data Mining*, vol. 5, no. 1, pp. 137-147, March 2017.

کنترل و هدایت خودکار ربات متحرک با استفاده از روش‌های یادگیری ماشینی

سمیه قندی* و هادی مختاری

گروه مهندسی صنایع، دانشکده مهندسی، دانشگاه کاشان، کاشان، ایران.

ارسال ۲۰۲۱/۱۰/۱۳؛ بازنگری ۲۰۲۲/۰۱/۳۱؛ پذیرش ۲۰۲۲/۰۳/۱۳

چکیده:

در بسیاری از کاربردهای رباتیک، ربات متحرک بایستی برای انجام کارها از یک منبع به یک مقصد بخصوص هدایت گردد. کنترل و هدایت اتوماتیک و موثر ربات متحرک یک حوزه چالش برانگیز در زمینه تحقیقات رباتیک می‌باشد. بنابراین در این کار، کنترل و هدایت اتوماتیک ربات متحرک با استفاده از روش های مختلف یادگیری ماشین مورد مطالعه قرار گرفته است. کنترل ربات متحرک این است که به ربات کمک شود تا تصمیم صحیح را در خصوص تغییر جهت بر اساس اطلاعات خوانده شده توسط سنسورهایی که در اطراف کمر ربات قرار دارند، اتخاذ نماید. روش های یادگیری ماشین با استفاده از ۳ مجموعه داده بزرگ اطلاعات خوانده شده توسط سنسورها و به دست آمده از پایگاه داده یادگیری ماشین UCI آموزش می‌بینند. روش‌های مورد استفاده شامل (i) متمایزکننده‌ها: جداساز ابرمکعب حریصانه و ماشین‌های بردار پشتیبان، (ب) رویکردهای پارامتری: جداساز بیز ساده با استفاده و بدون استفاده از روش‌های کاهش تعداد ویژگی، (iii)الگوریتم‌های شبه پارامتریک: الگوریتم EM، C-means، K-means، خوشه‌بندی Agglomerative، (IV) رویکردهای ناپارامتریک برای تعریف تابع چگالی: هیستوگرام و برآوردکننده‌های Kernel، (v) رویکردهای ناپارامتریک برای یادگیری: k نزدیکترین همسایه و درخت تصمیم، و (vi) ترکیب چندین یادگیرنده: Boosting و Bagging می‌باشند. این روش ها بر اساس معیارهای مختلف با یکدیگر مقایسه گردیده اند. نتایج محاسباتی بیانگر عملکرد بهتر روش های پیاده شده نسبت به روش های قبلی معرفی شده برای مجموعه داده مذکور می باشد. به طور کلی Boosting، Bagging، درخت هرس نشده و درخت هرس شده (θ= 10-7) در مقایسه با موارد موجود نتایج بهتری به همراه داشته است .همچنین کارایی درخت تصمیم پیاده‌سازی شده بهتر از سایر روش‌های بکار گرفته شده است و این روش دقت طبقه‌بندی، نرخ TP، نرخ FP و MSE کلاس‌ها را به میزان ۰,۱، ۰,۱، ۰,۰۰۱ و ۰,۰۰۱ درصد بهبود می‌بخشد.

کلمات کلیدی: هدایت ربات متحرک، جداساز، رویکرد پارامتریک، رویکرد شبه پارامتریک، رویکرد ناپارامتریک.