HIERARCHICAL BEHAVIOR CATEGORIZATION USING CORRELATION BASED ADAPTIVE RESONANCE THEORY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA YAVAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

SEPTEMBER 2011

Approval of the thesis:

HIERARCHICAL BEHAVIOR CATEGORIZATION USING CORRELATION BASED ADAPTIVE RESONANCE THEORY

submitted by **MUSTAFA YAVAŞ** in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Engineering Department, Middle East Technical University by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Ferda Nur Alpaslan _____
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Varol Akman
Computer Engineering Dept., Bilkent University _____

Assoc. Prof. Ferda Nur Alpaslan
Computer Engineering Dept., METU _____

Assoc. Prof. Tolga Can
Computer Engineering Dept., METU _____

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Dept., METU _____

Asst. Prof. Pınar Şenkul
Computer Engineering Dept., METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MUSTAFA YAVAŞ

Signature :

ABSTRACT

HIERARCHICAL BEHAVIOR CATEGORIZATION USING CORRELATION BASED ADAPTIVE RESONANCE THEORY

Yavaş, Mustafa
Ph.D., Department of Computer Engineering
Supervisor : Assoc. Prof. Ferda Nur Alpaslan

September 2011, 92 pages

This thesis introduces a novel behavior categorization model that can be used for behavior recognition and learning. Correlation Based Adaptive Resonance Theory (CobART) network, which is a kind of self organizing and unsupervised competitive neural network, is developed for this purpose. CobART uses correlation analysis methods for category matching. It has modular and simple architecture. It can be adapted to different categorization tasks by changing the correlation analysis methods used when needed.

CobART networks are integrated hierarchically for an adequate categorization of behaviors. The hierarchical model is developed by adding a second layer CobART network on top of first layer networks. The first layer CobART networks categorize self behavior data of a robot or an object in the environment. The second layer CobART network receives first layer CobART network categories as an input, and categorizes them to elicit the robot's behavior with respect to its effect on the object. Besides,

the second layer network back-propagates the matching information to the first layer networks in order to find the relation between the first layer categories.

The performance of the hierarchical model is compared with that of different neural network based models. Experiments show that the proposed model generates reasonable categorization of behaviors being tested. Moreover, it can learn different forms of the behaviors, and it can detect the relations between them. In essence, the model has an expandable architecture and it contains reusable parts. The first layer CobART networks can be integrated with other CobART networks for another categorization task. Hence, the model presents a way to reveal all behaviors performed by the robot at the same time.

Keywords: Robot behavior categorization, machine learning, adaptive resonance theory

ÖZ

İLİNTİ TEMELLİ UYARLANIR REZONANS KURAMI KULLANARAK SIRADÜZENSEL DAVRANIŞ SINIFLANDIRMA

Yavaş, Mustafa
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ferda Nur Alpaslan

Eylül 2011, 92 sayfa

Bu tez davranış tanıma ve öğrenmede kullanılabilecek yeni bir davranış sınıflandırma modeli sunar. Bu amaçla, kendini örgütleyen, deneticisiz, rekabete dayalı yapay sinir ağı olan İlinti Temelli Uyarlanır Rezonans Kuramı (CobART) ağı geliştirilmiştir. CobART sınıf eşlemede ilinti çözümleme yöntemlerini kullanır, birimsel ve basit bir mimariye sahiptir. Bu yapı, kullanılan ilinti çözümleme yöntemleri gerektiğinde değiştirilerek farklı sınıflandırma görevlerine uyarlanabilir.

CobART ağları davranışların uygun bir şekilde sınıflandırılması amacıyla sıradüzensel bir şekilde birleştirilmiştir. Sıradüzensel yapı, birinci katmandaki CobART ağlarının üzerine ikinci katmanın eklenmesi ile geliştirilmiştir. Birinci katmandaki CobART ağları robotun veya ortamdaki nesnelerin davranışını sadece kendi verilerini kullanarak sınıflandırır. İkinci CobART ağı katmanı, birinci katmandaki CobART ağlarında oluşan sınıfları girdi olarak kullanır ve bunları sınıflandırarak robot davranışını nesne üzerindeki etkisine göre ortaya koyar. Bunun yanında, ikinci katmanda oluşan eşleme

bilgisi birinci katmandaki ağlara geri yansıtılarak, bu katmanda üretilen sınıflar arası ilişki hesaplanır.

Sıradüzensel modelin başarımı yapay sinir ağlarını kullanan çeşitli modeller ile karşılaştırılmıştır. Test sonuçları gösteriyor ki; önerilen yöntem teste tabi davranışlar için makul bir sınıflandırma üretmektedir. Bunun yanında bu yöntem, davranışların çeşitli biçimlerini öğrenebilmekte ve bunlar arasındaki ilişkiyi belirleyebilmektedir. Esas itibariyle bu yöntem genişleyebilir bir yapıya sahiptir ve yeniden kullanılabilir parçaları içerir. Birinci katmandaki CobART ağları farklı bir sınıflandırma görevi için başka CobART ağları ile birleştirilebilir. Sonuç olarak bu yöntem, robot tarafından aynı anda yapılan tüm davranışları ortaya çıkaran bir yol sunar.

Anahtar Kelimeler: Robotlarda davranış sınıflandırma, makine öğrenimi, uyarlanır rezonans kuramı

To the next generation of my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Assoc. Prof. Ferda Nur Alpaslan for her guidance and efforts. Without her help and motivation, it would be so hard for me to come to the end.

In addition, I would like to express my appreciation to Prof. Dr. Varol Akman and Assoc. Prof. Tolga Can for their suggestions and supports at the thesis committees.

I offer special thanks to my friend Seyit Sabri Seyhan for his contributions on the grammatical and the semantic checks of the papers. Besides, thanks to him and to another friend of mine Zafer Ataser for their supports and for sharing the stress.

Thanks to ETC, for their tolerance on work hours, when I need to study.

At last but not the least one, I would like to express my gratitude to my family for their support through my education life. Besides, special thanks to my mother Nevin Yavaş who influenced the importance of the education to me starting from my childhood.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Different types of robots are taking part more and more everyday in daily life [61]. Studies on human- or animal-like robots have been carried out to build robots that can perform some complex tasks by interacting with humans in dynamic environments. Preliminary versions of these robots can perform different tasks but they are having assumptions over the environment, over the learning method, or over the task.

The aim of this study is to categorize behaviors in order to elicit alternative forms of primitive behaviors and to find relations between them. This process can help the construction of a base for behavior learning and generation, and it may be used for behavior recognition of objects moving around. To this end, it may be good to start with the definition of behavior. Although there is no common definition for behavior, it is generally defined as a reaction to a stimulus [63]. This study considers behavior as a time-extended action that achieves a set of goals [32] and causes changes in the environment or to the robot itself.

The idea of developing a generic, unsupervised and self-organizing behavior model motivated us to focus on the behavior categorization. Thereby, behavior categorization by ART 2 (Adaptive Resonance Theory) network was implemented first [61]. ART 2 [4] introduces a method of self-organization and stable category recognition via unsupervised competitive neural networks, which is used to categorize arbitrary sequences of continuous-valued input patterns. The experiments showed that behaviors are not stabilized even after long-time learning, and some similar behaviors are matched with different categories, and unrelated data are grouped within the same categories [63].

To increase the categorization performance of ART 2, a priori SOM (Self-Organizing Map) categorization was combined with ART 2 network. Thus, topological representation of the behavior data by SOM can result in better categorization by ART 2. SOM is a competitive learning method based on feature mapping [61]. The main idea behind the SOM method is arranging the location of output units on the plane in a way that it reflects the input topology in the best way [21, 33]. The experiments over SOM and ART 2 [61] showed that using a priori SOM categorization didn't contribute to the ART 2 categorization results.

As a result, a new type of unsupervised, self-organizing stable category generation network was developed to use for behavior categorization. The main contribution of this network relies on the correlation analysis methods used for category matching [62]. Therefore it is named as CobART. CobART network is inspired from ART 2 network, but has more modular and simpler architecture that can be easily adapted to different domains. CobART performs satisfactory categorization for the domain where patterns are constructed from consecutive continuous-valued inputs [61].

The behavior categorization results by CobART network showed that the adequacy of CobART categorization is more reasonable than ART 2, even when priori SOM categorization is used. However, it can be observed from the experiments that CobART itself is not enough for behavior categorization [63]. CobART discriminates different behaviors but it may generate more than one category for similar behaviors. Generation of multiple categories for the similar behaviors does not imply inadequacy, because it is also a need to learn different forms of the behaviors. Nevertheless, the relation between these categories shall be revealed for correct behavior recognition and learning tasks. Therefore, it was concluded that there is a need for a method that can differentiate fast and slow motion of the robot, and can understand that both of them are types of the similar behavior. Hence, hierarchical behavior categorization model is developed.

Hierarchically integrated CobART network is constructed by adding a second layer CobART network that applies extra categorization over the first layer networks' category outputs [63]. Thus, robot motion, object motion and distance data are categorized with different CobART networks at the first layer, and category outputs of these

networks are fed into the second layer CobART network as an input resulting in second categorization process. Besides, correlation scores between the categories, and category matching information from the second layer are used to calculate the category similarity. The experiments over the hierarchical model showed that the new model outperforms the single categorization by CobART. The model has an expandable architecture and it contains reusable parts. The first layer CobART networks can be integrated with other CobART networks for another categorization task.

The main contributions of this study are as follows:

- CobART, a new type of unsupervised, self-organizing stable category generation network is presented, and its success is compared with that of ART 2. Experiments show that, CobART generates reasonable categorization for consecutive continuous-valued input patterns.
- CobART has a modular and simple architecture that can be easily adapted to different domains. Correlation analysis methods used in category matching can be changed easily with the domain specific methods when needed.
- Hierarchical behavior categorization model is introduced. The model uses two layer CobART networks for reasonable categorization of behaviors. The proposed model can elicit different forms of the primitive behaviors, and can find the correlation between them.
- Hierarchical model does not perform categorization for one specific behavior, but can reveal different behaviors performed by the robot at the same time. While a first layer CobART network in the model can be processing gripper motion data of the robot, another network in the same layer can be categorizing object behavior. At the same time, a network in the second layer can be categorizing the behavior of the robot by analyzing its effect on the object staying in the environment.

The proposed model is tested by *Khepera* robot [29] using *Webots* simulator [38]. *Pushing*, *approaching*, and *striking* behaviors are selected for testing purpose [63]. The simulator was programmed for each behavior to collect *Khepera*'s own sensory

inputs as a training and test data. Multiple runs were performed and the motion data was logged for each run. The robot was made to move from various distances with varying velocities in order to collect various data.

The results of these studies have already been published in [61], [62] and [63]. Therefore, the significant parts of the explanations in this and the following sections are taken from these publications.

The remainder of this thesis is organized as follows. The related studies on behavior recognition and learning are presented in Chapter 2. Following that, background information about correlation analysis methods, and ART 2 and SOM networks are summarized in Chapter 3. Chapter 4 explains CobART network in detail and compares its performance with that of ART 2. Then, Chapter 5 presents alternative behavior categorization models and experimental evaluation of them. At last concluding remarks are reported in Chapter 6.

CHAPTER 2

RELATED WORK

The studies on behavior recognition and learning are far from satisfactory levels [63]. The studies generally uses neural networks, probabilistic models, imitation learning, genetic algorithms, and non-learning models. Following papers may give a rough idea about the way these studies are carried out.

2.1 Studies with Neural Networks

Fox et al. [13] introduced a machine learning approach that elicits internal steps of the navigation behavior from raw sensor data using Kohonen Networks (see Section 3.3) and Hidden Markov Model (HMM)(see Section 3.5). Kohonen Network is used for clustering the feature vectors which are combination of finite sequence of observations. The outputs of the clustering process are used at the HMM for state transitions. For the determination of the states of the HMM, labeling by human operator is used. Human observable general states are selected as a priori states such as *start*, *end*, *searching*, *hesitating*, *progressing*, and *obstacle avoiding*. Throughout the observation period, operator labels the behavior of the robot using priori states. Then, feature vectors are associated with the last seen labels. But, behavior can not be expressed with these general labels. Because, the aim is revealing hidden states of the behavior. State splitting strategy is used for division of the states in a way that, characteristic vectors of all clusters having association with selected state are drawn as a graph, and maximal clique[1] within these graphs are selected as new substates.

[1] Maximal clique is a "clique that is a complete subgraph and not a subgraph of a larger clique" [39].

Then, Expectation Maximization[2] algorithm is used for parameters re-estimation of the HMM. In order to evaluate the quality of the learned HMM the sequence of states are compared with human observed states. Viterbi algorithm is used in order to find most probable sequence of the states for a given trajectories on a learned model to compare with human observed states. On the other hand, quality of the HMM is also evaluated by investigating the stability of the clustering for different sizes and random initialization.

Tani [56] presents a novel hierarchical neural network for behavior learning. Model includes two layer Recurrent Neural Network (RNN) (see Section 3.4) running on different time scales. The lower level RNN learns primitive behaviors by using sensory and motor data. The sensory and motor outputs of the network are fed into the inputs at the learning phase in order to use the past information. On the other hand, the top level RNN learns the sequences between the primitive behaviors and time spent at each of them. The learning occurs based on the interaction between bottom-up and top-down signals. Mismatch between top-down commands and sensory-motor data are back-propagated to the lower level RNN. In the learning phase, seven primitive behaviors are defined firstly. Then, some combinations of them are generated through manual guidance of the operator. The lower level RNN is trained with these sequences. When these behaviors are learned, three new behavioral patterns containing different sequences of primitive behaviors are used to test model if it can learn novel behaviors. The test results show that, the robot can regenerate learned behaviors without significant discrepancies when order of the primitive behaviors selected according to design rules.

Moreover, Paine and Tani [51] proposed a model based on two-layer recurrent neural network that learns generation of different navigation tasks. Lower layer learns basic sensory-motor primitives such as obstacle avoidance, turning left/right and moving straight through the corridor using sensory inputs and generating motor commands. Higher layer network learns sequences of the primitive behaviors. Higher layer does not have direct association between sensory inputs and motor outputs. Instead, it has

[2] Expectation Maximization algorithm "iteratively computes maximum-likelihood estimation when the observations can be viewed as incomplete data. Since each iteration of the algorithm consists of an expectation step followed by a maximization step it is called as EM algorithm" [11].

two special neuron named as *control neurons* those have associations with the neurons in the lower layer. Control neurons effect the generation of the motor primitives. On the other hand, higher layer has two other special neurons named as *task neurons*. Initial values of the task neurons determine the task that will be implemented. Genetic algorithm is used in order to learn weights of the associations. Experiments show that after enough training, desired tasks can be implemented by setting corresponding initial value to the task neurons. But, when tests are done in an enlarged environments by expanding corridors, the results were not promising.

Fung and Liu [15] worked on robot behavior learning that uses a specific neural network named as Behavior Learning/Operating Modular (BLOM) architecture which is formed by the integration of the ART networks and associative memories. Sensory inputs are categorized into S-categories and action outputs are categorized into A-categories using Fuzzy ART networks (see Section 3.2). Associative memories are used for establishing association between the S-categories and A-categories. Some modifications are made on the ART network in order to find best vigilance parameter. Vigilance parameter adaptation is made according to game-theoretic approach. Wall-following behavior learning is tested on the proposed model. The results shows that, robot cannot keep distance to the wall in a constant value, but can follow the wall.

Gu and Su [17] present categorization of joint actions using ART 2 networks. A separate ART 2 network is dedicated for each joint action. Generated categories are interconnected with behavior labels uttered by the instructor. Experiments gave promising results.

Blynel and D. Floreano [2] worked on Continuous Time Recurrent Neural Network (CTRNN) and Plastic Neural Network (PNN), and compared their capabilities on behavioral tasks. Recurrent neural networks can detect time dependent activation patterns from sensors and other neurons. In CTRNN, time dependent information is kept in neurons, whereas it is kept in the connection strengths for PNN. Another difference between these two networks is that sensory receptors in CTRNN receive information only from sensors whereas sensory neurons in PNN receive information from all neurons in the networks including other sensory neurons. The networks are trained using genetic algorithm in order to learn a behavior of staying in the reward

area at maximum time. CTRNN network gave better results for this test. When adaptation to changes is tested, performance of the CTRNN decreases much more than PNN. Especially when sensory-motor adaptation is required, PNN performed better than CTRNN. This is because, PNN rapidly changes weight values when sensory information changes significantly.

Itoh et al. [25] introduced specific neural networks designed for storing different patterns, inspiring from associative memory model[3]. Networks having association between stored patterns are made coupled by connecting neurons being in the same position. When this is done, networks can retrieve the patterns from coupled network. Using this model, human memory can be simulated. Like a human memory that can remember different things in the same physical condition depending on their certain mood, coupled neural networks can success this for robots. The model is developed for expressing robots mood continuously while it has a natural communication with human.

2.2 Studies with Probabilistic Models

Infantes et al. [23] proposed a generic model for learning the behavior model of the robot for a given task, using observation data. Based on the learned model, the study presents a way to improve the robot behavior according to environment parameters and user preferences. The model is developed by using Dynamic Bayesian Network[4] and Dynamic Decision Network[5], and is tested with navigation task.

Han and Veloso [19] present high-level robotic behavior recognition from low-level sensor inputs using revised Hidden Markov Model. Reject states are added to the classical HMM in order to stop processing of HMM when observation does not conform to the definition. The states of the model are defined as a priori information.

[3] Associative memory models are designed based on neural networks to store patterns. The stored patterns can be generated as an output when related inputs are fed into the network [20].

[4] Dynamic Bayesian Network is a kind of stochastic model. "It introduces a state space structured into controllable parameters, environment variables, robot state variables and mission variables, with explicit causal links between these variables, within each state and from state to state" [23].

[5] Dynamic Decision Network is similar to Dynamic Bayesian Network except the "transitions are labeled with cost or rewards" and "some variables are said controllable, and algorithms for decision making give values to this controllable variable in order to maximize utility" [23].

Robotic soccer game is selected to use the model for the recognition of opponent's behaviors and for narrating the game. A specific HMM is defined for each behavior.

Morisset and Ghallab [45] introduce a model that learns performing appropriate skills to accomplish a given task. The model contains a number of built in sensory-motor functions. A set of skills using these functions are defined by Hierarchical Task Networks (HTN). "The objective of an HTN planner is to produce a sequence of actions that perform some activity or task. The description of a planning domain includes a set of operators similar to those of classical planning, and also a set of methods, each of which is a prescription for how to decompose a task into subtasks" [46]. The model learns which skill to apply by using Markov Decision Process. Markov Decision Process is a kind of stochastic model that, it has finite set of states and actions. Once an action is performed, new state is determined according to probabilistic distribution function, and reward is determined according to reward function.

Kelley et al. [31] present a method for intent recognition of the agents from robot's perspective. To do that, Hidden Markov Models are constructed for each behavior, and these models are trained by applying the behaviors with the robot. After learning is completed, the robot can be used as an observer to calculate necessary parameters from the agent's perspective in order to recognize agent's intent as early as the activity is started. The experiments show that, the model generates high accuracy rates and early detection durations.

Inamura et al. [22] worked on mimesis that is one of the framework models in cognitive psychology which explains how human learns behavior by observation of other behaviors. According to proposed approach, sample behaviors are abstracted into motion elements and then sequence of the motion elements are learned using Hidden Markov Model. Motion elements correspond to joint angles for a period of time. For the motion generation, maximum probability for the defined symbol sequence is searched. Results show that, satisfactory motion can be generated, and others' motion can be recognized as well.

Osentoski et al. [50] present capability of 2-level Abstract Hidden Markov Model (AHMM). "The AHMM is a multi-scale statistical model for representing behav-

iors in stochastic, noisy situations. It provides a method of modeling hierarchical behaviors" [50]. The structure and the parameters of the model is explained in detail. Besides, learning of the parameters based on the raw sensor data for navigation behavior at indoor environment using Expectation Maximization is explained. The model includes states, observations, mixture components at the lowest level, and behavior and termination flag at each upper level. The model is compared with 1-level AHMM. Test results show that, 2-level AHMM performance is better especially at the environment where training data is generally similar, and hence classification is difficult.

Liao et al. [37] introduced a model based on Hidden Markov Model to learn and infer daily movement of the subject using raw GPS (Global Positioning System) data via unsupervised way. A model consists of three levels. Lowest level contains GPS sensor measurement nodes, transition nodes, a node for position and velocity of the subject and location of the subject's car. Middle level has transportation mode nodes, trip segment nodes and trip switching and mode switching nodes. Top level has goal nodes and goal switching nodes. Trip segments corresponds to sub-segments of the goal. Transportation nodes may have a value from bus, foot, car, building set. Goal states are defined as the locations where subject spend much time. Bus stops, parking lots, etc. are set as mode transfer locations. Expectation Maximization is used to learn these nodes and to learn parameters between goal segments and between trip segments. Experiments show that the proposed model generates satisfactory results. The model can even determine deviation from the goals. Thus, it can help to cognitively-impaired individuals as a personal guidance for safely moving.

2.3 Studies with Imitation Learning

Kubota [34] presents interactive learning between instructor and a robot based on imitation. In this study, Spiking Neural Networks[6] are used for extracting spatial and temporal information of human gesture patterns. Self-organizing maps are used

[6] Spiking Neural Networks are "known as the third generation of neural network models. They are more closely related to their biological counterparts compared to previous neural networks. These networks employ transient pulses for communication and computation" [18].

for gesture classification, while a steady-state genetic algorithm[7] is used for action pattern generation with respect to human gestures. It is assumed that, the instructor repeats same gesture when he can not correlate his gestures with the robot's action. When he is satisfied with robot's action, he will make some small differences in his gestures.

Ito and Tani [24] present a novel imitative model that can learn several movement patterns and associates them with the human demonstrator's corresponding hand movements. The model is developed by using recurrent neural network with parametric bias (RNNPB). "RNNPB is a version of the RNN where the parametric bias units allocated in the input layer play the roles of mirror neurons, since their values encode both of generating and recognizing the same movement patterns" [24]. In the learning, different parametric bias values assigned for each movement patterns. The experiments show that, robot can generate correct movement patterns synchronously with respect to human hand motions. When new hand motions are demonstrated during the testing, robot cannot follow them but generates a various movement patterns.

Borenstein and Ruppin [3] present a specific framework for imitation learning. The proposed approach consists of a specific neural network and evolution using genetic algorithms. Environment data and demonstrator's actions are inputs to the neural network while motor commands are the outputs. Genetic algorithm is applied on the network parameters. The fitness of the individuals are calculated based on the properness of the actions that they performed depending on the state. Results shows that after 2000 generation, the evolved agents can perform proper actions.

Billing and Hellstrom [1] worked on behavior recognition using Learning from Demonstration (LFD) techniques. They introduce β-Comparison, AANN-Comparison (Auto-associative Neural Networks Comparison) and S-Comparison techniques for this purpose and compares their performance. β-Comparison is based on action comparison. It assumes two skills are equal if they generate similar actions for the given sensory inputs. AANN-Comparison uses regular feed-forward neural networks. This network is trained to learn input values to be mapped by the same values at the output

[7] Steady-state genetic algorithm is a kind of genetic algorithm that, only a few individuals of the population is replaced with new individuals at each iteration [34].

layer. S-Comparison is "a prediction-based control algorithm inspired by the human neuromotor system" [1]. Five different types of the skills are demonstrated to the alternative techniques. Then, their performance are evaluated by two different test cases each containing some combination of the already learned skills. The results show that S-Comparison outperforms other techniques. β-Comparison generates the worst performance. It may be the consequence of only using action vector at the skill comparison.

Palm and Iliev[52] worked on grasp recognition task. Robot collects motion data of the grasp behavior demonstrations. Then, these data are segmented into the clusters. However, the study does not focus on the segmentation task, and assumes the segmentation has already been performed. Three different recognition methods are presented and their performance are compared for the recognition task. First method uses distance between time clusters for recognition purpose. Second method is named as qualitative fuzzy recognition rules. In this method, combination of the test grasps are compared with the grasp models and their distance norms are calculated at start. Then, the vectors corresponding to minimum and maximum points of the comparisons are determined. Following that, a set of fuzzy rules are used to decide if special combination of these vectors belong to the model grasps. At end, similarity degrees between the model grasps and combination of test grasps are computed. The third method is based on HMM. Five states are selected first: *open hand*, *half open hand*, *middle position*, *half closed position*, and *closed hand*. The clusters from the sampled motion data are used as an observation items of the model. Finally, recognition of fifteen different object grasps are tested. Test results show that first method outperforms the others, while the third method becomes the weakest one.

In addition, Palm et al. [53] present a way of modeling and recognizing robot skills. In order to do that, robot captures the motion data first, while the demonstrator performs the target skills. The training of the system is performed by segmentation of skills into the phases. Then, phases are modeled using fuzzy time modeling method. Details of this method is explained in this paper. The study is tested using contour following skills and satisfactory results are obtained.

2.4 Other Studies

Oates et al. [48] worked on a method for clustering experiences of a robot in a complex environment based on dynamic sensory data. The motivation of the study is, robots should know the effects of their action over the environment, in order to act around the dynamic environment. Dynamic Time Warping (DTW)[8] method is used to measure similarities between the experiences. The action length within the experiences is a big problem at the comparisons. DTW proposes solution for this case by moving the backward/forward at the time domain. Clusters are constructed according to minimum distance criteria. The prototypes of the clusters are selected either with same criteria or as a result of averaging. The proposed method is evaluated with respect to clustering made by human. The results are satisfactory such that, 82-100% accordance levels are reached.

Peula et al. [54] present a supervised reactive behavior learning model. In the training phase, a human supervisor controls a test robot to perform target tasks while the robot collects sensory data and corresponding motion commands to learn their association using Case Based Reasoning[9] technique.

Yang and Li [60] worked on a model that is designed using subsumption architecture. Fuzzy logic control is applied to the obstacle avoidance behavior. The proposed model uses genetic algorithm for vision based landmark recognition.

Lebeltel et al. [36] studied Bayesian Robot Programming (BRP) and worked on a method to implement different types of robot behaviors. The method uses Bayesian inference[10] rules. The problem is defined according to specific formulas based on related sensory data, defined variables and preliminary knowledge. The results are satisfactory both for basic behaviors and complex behaviors. The robot is programmed to extinguish fires when it detects it, and to go to charging at energy is decreased.

[8] "Given two experiences, E_1 and E_2 (more generally, two continuous multivariate time series), DTW finds the warping of the time dimension in E_1 that minimizes the difference between the two experiences" [48].

[9] A powerful strategy to use in expert systems is "reasoning from cases, examples of past problems and their solutions. Case Based Reasoning uses an explicit database of problem solutions to address new problem-solving situations" [39].

[10] "In general, a Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables" [44].

Nicolescu and Mataric [47] present hierarchical architecture for behavior based system. It introduces abstract and primitive behaviors, and relations between them, and relation between abstract behaviors. The architecture is expandable for addition of new tasks and behaviors. Although the architecture is interesting, no learning method is used.

In addition, Koenig and Mataric [32] present demonstration based behavior learning. In the study, human teacher has a key role at selection of features when determining preconditions and postconditions for primitive behaviors. This paper does not propose a learning method, but holds a case list to keep cases corresponding to precondition and postconditions, and an action to reach to the postcondition. In order to perform a task, a plan is generated using case list, and start and goal states.

Kanda et al. [30] introduces a humanoid robot that is developed based on constructive approach, bottom-up design. Although evaluations gave satisfactory results for the behaviors and the gestures of the robot, none of the machine learning approach is used for the development and evaluation.

CHAPTER 3

BACKGROUND

The studies on behavior categorization is started by developing ART 2 network for the categorization task. ART 2 is selected because of its self-organized and unsupervised nature. Later, priori SOM categorization is combined with ART 2 categorization, having the idea that topological representation of the behavior data by SOM can result in better categorization by ART 2 [61]. Following that, a new type of unsupervised, self-organizing stable category generation network was developed to use for behavior categorization. The main contribution of this network relies on the correlation analysis methods used for category matching. Following subsections present detailed information about ART 2 and SOM networks, and correlation analysis methods. Moreover, an introductory information about Recurrent Neural Networks and Hidden Markov Model are presented. These models are not used within this study, but they have been used by the multiple studies those are explained in the previous chapter.

3.1 Correlation Methods

Correlation means "interdependence between two sets of numbers' ' [27]. In addition, it is also defined as a measurement of an association between two or more variables or data sets. Correlation analysis results in three types of association [58]:

- Positive correlation: Two variables are increasing or decreasing accordingly. Positive correlation scores can have the values in the range of [0:1].

- Negative correlation: When one of the variable is increasing, the other one decreases, or vice versa. Negative correlation scores can have the values in the range of [-1:0]
- No correlation: It is also known as zero correlation. When one the variable is changing, the other one stays constant. In this case, correlation score is equal to zero.

In this study correlation of two vectors[1] will be investigated. These vectors will contain different instances of the same variables. Thus, correlation analysis methods will be used to measure how similar two vectors are.

In fact, there is no common definition for similarity. In mathematics, two figures are defined as a similar if their corresponding angles are equal, and their corresponding sides are proportional [27]. When deciding on the similarity, the important problem is determination of correct features that makes them similar. In [59], Medin and Ortony are suggested to start by using directly recognizable features, which are named as "surface properties" when deciding on the similarity. After that, when "deeper conceptual knowledge" are revealed, they can be added to the similarity model depending on their association to the concept. In this study, the changes on the vector shapes and a distance between the vectors are the candidate features when determining the similarity.

Alternative correlation analysis methods will be introduced in the following subsections. Figure 3.1 shows sample vectors tested by these methods. The resultant correlation scores for each method are presented in Table 3.1.

3.1.1 Euclidean Distance Method

The well-known Euclidean Distance Method (EDM) measures the distance between two vectors as in (3.1) [62]. In the equation, it is assumed that, sizes of the vectors are the same, and the values of the vectors *w1*, and *w2* are in the range of [-1:1].

[1] The meaning of the *vector* in this study is not a quantity with a magnitude and a direction, but is a one-dimensional array. It does not have to contain linear data.

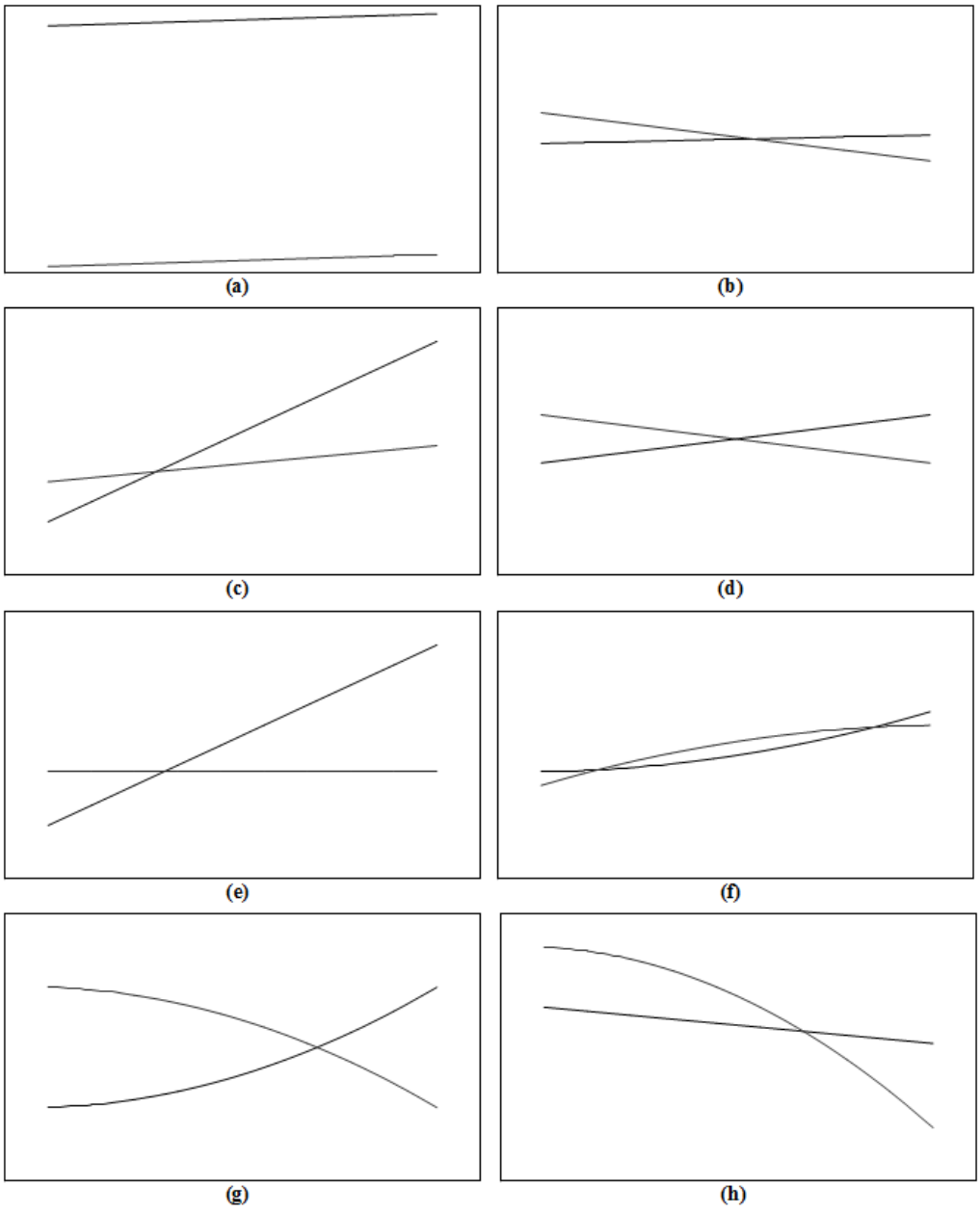

Figure 3.1: Sample vectors for correlation analysis methods

Table 3.1: Correlation analysis methods' measurements for sample vectors

Samples	EDM	CWA	DCM	BAM	DAM
(a)	-0.8	1	1	1	1
(b)	0.86	-1	-0.59	-1	-1
(c)	0.58	1	0.6	1	1
(d)	0.77	-1	-1	-1	-1
(e)	0.49	0	0	0	0
(f)	0.92	0.88	0.72	0.5	0.11
(g)	0.36	-1	-1	-1	-1
(h)	0.63	0.97	0.66	0.75	0.55

EDM generates matching score in the range of [-1:1]. Table 3.1 illustrates matching scores of EDM for sample vectors in Figure 3.1. It is clear from the figure that, vectors having short distance between them result in high EDM matching score, while vectors having long distance between them result in low matching score.

$$EDM(w1, w2) = 1 - \sqrt{\frac{\sum_{i=1}^{I}(w1_i - w2_i)^2}{I}} \tag{3.1}$$

In the equation, i^{th} elements of the vectors are represented with $w1_i$ and $w2_i$, and vector length is represented by I.

3.1.2 Correlation Waveform Analysis

Correlation Waveform Analysis (CWA) [57] measures the correlation of two same-sized vectors depending on the shape of change of the vectors [62]. CWA compares elements of two vectors, and generates matching score in the range of [-1:1]. Table 3.1 illustrates matching scores of CWA for sample vectors in Figure 3.1. Maximum matching is obtained when shape of change of the vectors with respect to their standard deviations are similar. In other words, vectors not have to be parallel to each other in order to have maximum CWA matching score. When two vectors are increasing/decreasing at the same rate, as in Figure 3.1(a) and Figure 3.1(c), "with respect

to their standard deviations" along their vector lengths, comparison of these vectors results in maximum matching.

$$CWA(w1, w2) = \frac{\sum_{i=1}^{I}(w1_i - \overline{w1})(w2_i - \overline{w2})}{\sqrt{\sum_{i=1}^{I}(w1_i - \overline{w1})^2 \sum_{i=1}^{I}(w2_i - \overline{w2})^2}} \tag{3.2}$$

Equation (3.2) represents the CWA calculation. In the equation, i^{th} elements of the vectors are represented with $w1_i$ and $w2_i$, and means of the vector data are referred as $\overline{w1}$ and $\overline{w2}$. Besides, vector length is represented by I.

3.1.3 Derivation Correspondence Method

Derivation Correspondence Method (DCM) is developed for behavior categorization task in order to check derivation similarity between two behavior data. DCM measures the correlation using instant derivation information [61]. Two vectors having the same size are compared if they are increasing/decreasing at the same points. The correspondence value is directly related with the magnitude of the instant derivation scale and the directions with respect to the starting point. Table 3.1 illustrates matching scores of DCM for sample vectors in Figure 3.1. Maximum matching is obtained when the vectors are parallel as in Figure 3.1(a).

$$DCM(w1, w2) = \frac{\sum_{i=1}^{I}\left(\frac{w1'_i}{w2'_i} + sign(w1_i - w1_1) \times sign(w2_i - w2_1)\right)}{2 \times I} \tag{3.3}$$

Equation (3.3) represents the DCM calculation. In the equation, $w1_1$ and $w2_1$ correspond to starting points of the vectors, and $w1'_i$ and $w2'_i$ represent instant derivations for i^{th} elements of the vectors. On the other hand, the division of $w1'_i$ over $w2'_i$ in the equation is reversed if the absolute value of the division is higher than one.

3.1.4 Bin Area Method

Bin Area Method (BAM) [57] [28] measures the correlation of two same-sized vectors depending on the comparison between the bins, where bin corresponds to a set of consecutive inputs. BAM compares bins of two vectors, and generates matching score in the range of [-1:1]. Table 3.1 illustrates matching scores of BAM for sample vectors in Figure 3.1, when 2-point bins are used.

$$BAM(w1, w2) = 1 - \sum_{i=1}^{B} \left| \frac{W1_i - \overline{W1}}{\sum_{k=1}^{B} \left| W1_k - \overline{W1} \right|} - \frac{W2_i - \overline{W2}}{\sum_{k=1}^{B} \left| W2_k - \overline{W2} \right|} \right| \tag{3.4}$$

Equation (3.4) represents the BAM calculation. In the equation, *W1* and *W2* corresponds to bin vectors for the *w1* and *w2* vectors. Besides, i^{th} elements of the bin vectors are represented with $W1_i$ and $W2_i$, and means of the bin vectors are referred as $\overline{W1}$ and $\overline{W2}$. In addition, bin vector length is represented by B, which is equal to vector length over the number of the points within the bins. On the other hand, the elements of the bin vectors are simply calculated as follows; $W1_1 = w1_1 + w1_2$ and $W1_2 = w1_3 + w1_4$ for the 2-point bins.

3.1.5 Derivative Area Method

Derivative Area Method (DAM) [57] measures the correlation of two same-sized vectors by using first derivation information of the vectors. DAM generates matching score in the range of [-1:1]. Table 3.1 illustrates matching scores of DAM for sample vectors in Figure 3.1.

$$DAM(w1, w2) = 1 - \sum_{i=1}^{I} \left| \frac{w1'_i}{\sum_{k=1}^{I} \left| w1'_k \right|} - \frac{w2'_i}{\sum_{k=1}^{I} \left| w2'_k \right|} \right| \tag{3.5}$$

Equation (3.5) represents the DAM calculation. In the equation, $w1'_i$ and $w2'_i$ represent first derivation of the i^{th} elements of the vectors, while I is used to represent vector lengths.

3.2 ART 2

ART 2 is a type of self-organizing, stable category recognizer and an unsupervised competitive neural network. It has mainly two subsystems; an attentional subsystem and an orienting subsystem [4]. Attentional subsystem contains a feature representation level F_1, a category representation level F_2, and bottom-up and top-down adaptive weights to encode activation patterns [62] as in Figure 3.2. F_1 contains six sublayers. All F_1 sublayers has one neuron for each dimension of the input vector and F_2 layer has one neuron for each category. There exist bottom-up and top-down adaptive filters between the pathways of F_1 and F_2. In ART networks [5], new inputs are encoded by changing the weights of a bottom-up adaptive filter. Top-down adaptive filters have a role of pattern matching and self-adjusting parallel search. On the other hand, orienting subsystem corresponds to resetting part, and becomes active when matching between bottom-up input to top-down category output has failed. In this case, alternative categories are tested until an adequate one is matched or a new category is established.

Figure 3.2 shows structure of the ART 2 network in detail. As shown in the figure, F_1 layer is divided into the six sublayers of w, x, u, u, p and q. The flow of data between these sublayers are indicated by the arrows. In addition, the inhibitory relations between these sublayers, including the nodes in the orienting subsystem, are indicated by the arrows tagged by the G node. The inhibitory signals carry magnitude of the input vector to their target nodes. These information are used for the normalization [26]. The calculations at these sublayers are presented in the following equations, from (3.6) to (3.12).

$$w_i = I_i + au_i \tag{3.6}$$

$$x_i = \frac{w_i}{e + \| w \|} \tag{3.7}$$

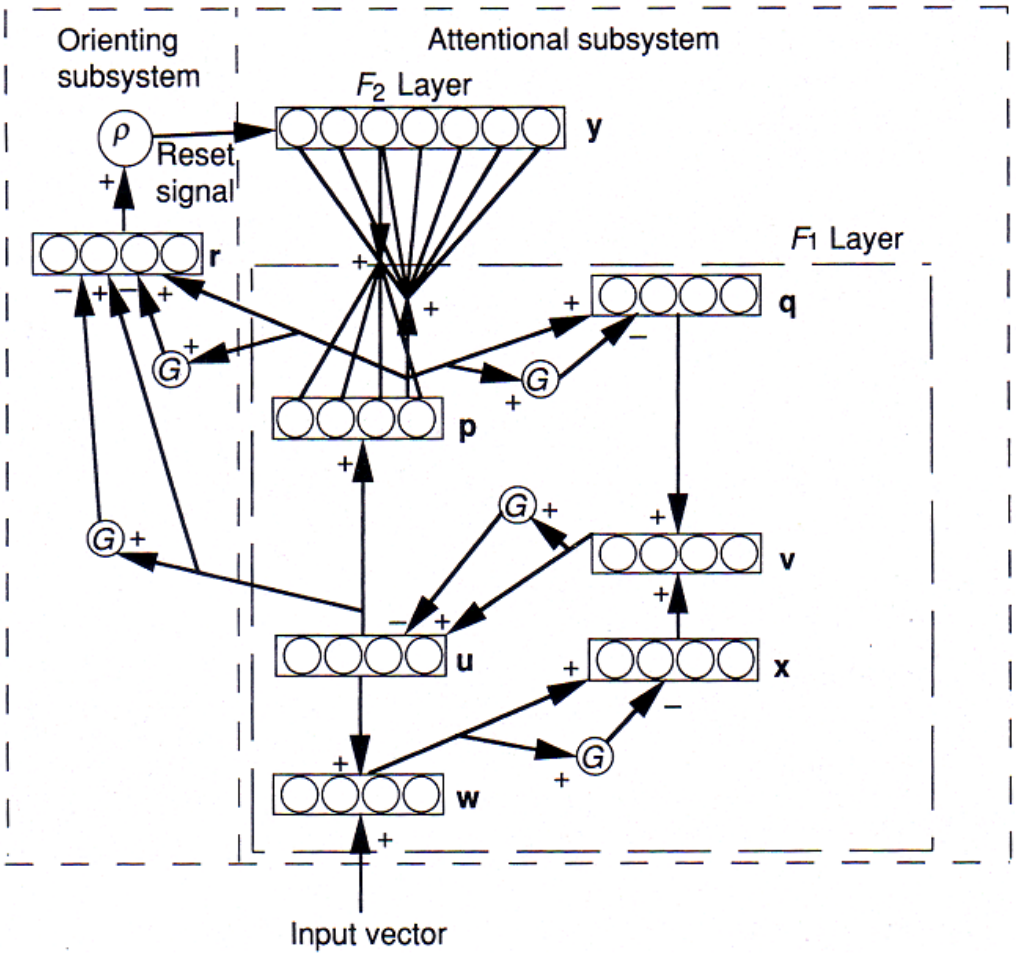

Figure 3.2: Structure of ART 2 network [26]

$$v_i = f(x_i) + bf(q_i) \tag{3.8}$$

$$u_i = \frac{v_i}{e+ \parallel v \parallel} \tag{3.9}$$

$$p_i = u_i + \sum_j g(y_j) z_{ji} \tag{3.10}$$

$$q_i = \frac{p_i}{e+ \parallel p \parallel} \tag{3.11}$$

$$f(x) = \begin{cases} \frac{2\theta x^2}{x^2+\theta^2} & 0 \leq \mathrm{x} < \theta \\ x & \mathrm{x} \geq \theta \end{cases} \tag{3.12}$$

In the equations, I corresponds to input vector, and z_{ji} corresponds to top-down weights. The parameters a, b, c, e, θ are constant values.

After input pattern is processed by the F_1 layer, categories in F_2 layers are checked to find maximum matching one according to (3.13) and (3.14). In the equations, z_{ij} corresponds to bottom-up weights, and J corresponds to index of the maximum matching category.

$$T_j = \sum_i p_i z_{ij} \tag{3.13}$$

$$T_J = max\{T_j\} \forall j \tag{3.14}$$

Equations (3.15) and (3.16) express *g(y)* function of the F_2 layer and its effect on the p sublayer of F_1. In the equations, d is constant parameter that is suggested to have a value within the range of [0:1].

$$g(y_j) = \begin{cases} d & \text{if } \mathrm{T}_j = \max\{\mathrm{T}_k\} \text{ where } \mathrm{k}^{th}\ \mathrm{F}_2 \text{ node} \\ & \text{has not been reset recently} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

$$p_i = \begin{cases} u_i & \text{if } \mathrm{F}_2 \text{ is inactive} \\ u_i + dz_{Ji} & \text{if the } \mathrm{J}^{th}\ \mathrm{F}_2 \text{ node is active} \end{cases} \tag{3.16}$$

Following that, top-down and bottom-up weights are updated according to (3.17) and (3.18), respectively.

$$\Delta z_{ji} = g(y_j)(p_i - z_{ji}) \tag{3.17}$$

$$\Delta z_{ij} = g(y_j)(p_i - z_{ij}) \tag{3.18}$$

At end, matching score between bottom-up input vector and top-down category templates are calculated, and compared with a vigilance parameter in the orienting subsystem. Vigilance parameter determines the quality of learned categories and supplies a balance between stability-plasticity parameters [62]. Learning system generates new categories or updates available ones for better categorization of each input to provide plasticity. On the other hand, the learned categories are stored against endless change, to enhance stability.

Equation (3.19) and (3.20) show matching score calculation and reset condition checks, respectively. If, the calculated score is less than the vigilance parameter ρ, reset is generated. Once reset is generated, alternative categories are tested until an adequate one is matched, or a new category is established when no matching is found.

$$r_i = \frac{u_i + cp_i}{e + \| u \| + \| cp \|} \tag{3.19}$$

$$\frac{\rho}{e+ \parallel r \parallel} > 1 \tag{3.20}$$

ART 2 parameters affect the quality of analog input pattern classification and there is no exact way of setting them [62]. Although this task is open for research, appropriate ranges have already been proposed [26] [16]. For the behavior categorization task using ART 2 in Section 5.2, a and b are set to 1.2, while c, e and θ are set to 0.1, 0 and 0.2, respectively. on the other hand, the suggested initial values for the top-down and bottom-up weights are represented in (3.21) and (3.22) respectively, where M represents the length of the input vector.

$$z_{ji}(0) = 0 \tag{3.21}$$

$$z_{ij}(0) \leq \frac{1}{(1-d)\sqrt{M}} \tag{3.22}$$

Continuous update of categories on ART 2 loses some information about old categories and it cannot protect stability of the categories adequately [62]. Besides, ART 2 focuses heavily on the significant input values when selecting a winner category. This feature is not enough for categorization of some type of patterns when they are constructed from consecutive input patterns and shape of input change is crucial for categorization. Moreover, ART 2 architecture is poor at dealing with noise [10], and there exist some uncertainties on when to apply weight update rule and how to apply reset mechanism.

3.2.1 Other ART Networks

Variety of ART networks exist. ART 1 neural network is introduced in [5]. ART 1 is very similar to ART 2, and is developed in response to arbitrary sequences of binary input patterns. In [8], ART 2-A is introduced which reproduces the behavior of ART 2 using more efficient algorithm and better noise tolerance. The results show

that ART 2-A and ART 2 generates similar categories over 50 analog input pattern categorization task [62]. In [9], Fuzzy ART, which is capable of rapid stable categorization of analog and binary patterns via incorporation of computations from fuzzy set theory, is introduced. Fuzzy ART and ART 2-A networks are compared in [14], and it is presented that Fuzzy ART is more sensitive to the noise and to the input representation order. Moreover, Fuzzy ART clustering is incoherent in input space and hence, ART 2-A better reflects the geometric relation of the input patterns to the categories.

Incorporation of supervision into the ART networks is introduced in [7] [6] [43]. ARTMAP [7] and Fuzzy ARTMAP [6] contain two ART networks and a map field that constructs association between the two ART network categories. SMART2 [43] uses class information of input patterns and enhances input patterns from different classes not to interfere in the construction and learning of the categories. SMART2 generates more reasonable categories than ART 2, but requires extra information for training data.

3.3 SOM

SOM is a competitive learning method based on feature mapping [61]. The main idea behind the SOM method is arranging the location of output units on the plane in a way that it reflects the input topology best [21] [33]. In order to do that, the values of closeness between output units are used. The outputs that are close to each other respond similarly to the same inputs. Generally, the input patterns are continuous valued and correspond to a point in the N-dimensional space. The output units are generally kept in the form of one or two-dimensional vectors. Figure 3.3 represents a sample schematic of the SOM network when two dimensional inputs are used, and the output units are kept in two-dimensional form.

Each output unit has a connection to the input layer having a weight vector at the size of the inputs' dimension. For instance, weight vector size is n for n-dimensional input space. No lateral connections between the output units are used, but neighboring

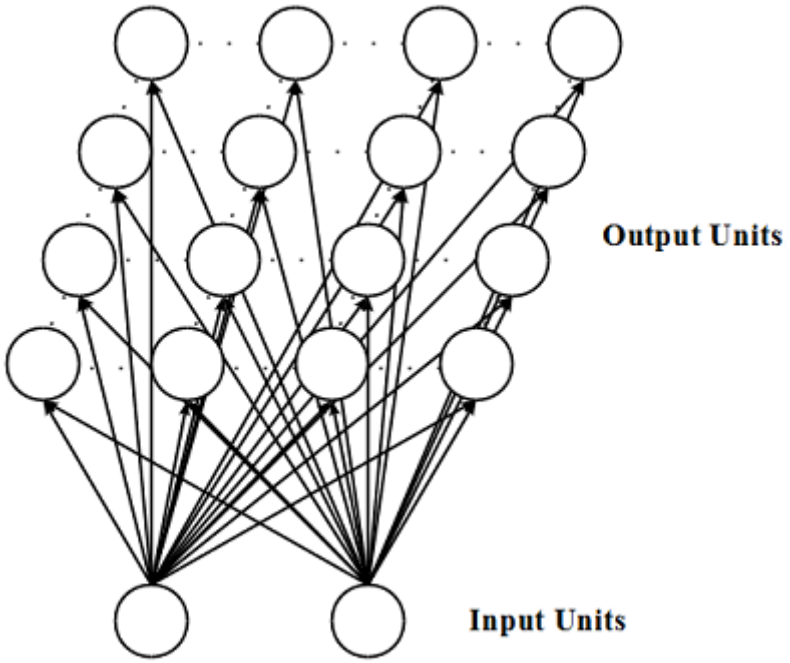

Figure 3.3: Schematic of SOM network

relation is established. In the figure, neighboring relation is represented by the black clipped lines. Whenever an input is fed in to the network, the winner output unit is determined by selecting closest output weight vector for that input vector using Euclidean distance as in (3.23) [21].

$$\left|w_j - E\right| \leq \left|w_i - E\right| \tag{3.23}$$

In the equation, E corresponds to the input vector E and w_i corresponds to weight vector for the i^{th} output unit. Besides, the winner output unit is represented by index j, while w_j is used to show its weight vector.

When the winner output unit is determined, the weights of the output units are updated with respect to (3.24), for all output units i and their weights on input dimensions k. The parameter η determines the learning rate of the algorithm. The update degree of the different output units is determined by the neighborhood function Λ [21]. The output units being closer to the winner output results in higher update rate.

$$\Delta w_{ik} = \eta \times \Lambda(i, j) \times (E_k - w_{ik}) \tag{3.24}$$

A candidate neighborhood function, is represented in (3.25). In the equation, r represents the position of the output unit, and σ represents the width parameter that ranges the neighborhood distance [21].

$$\Lambda(i, j) = e^{(-|r_i - r_j|^2/2\sigma^2)} \tag{3.25}$$

Learning rate and width parameters has high effect on the learning time and the quality of the mapping. Different numbers of tests for constant width parameter and learning rate showed that the network reaches an improper state in a short time, and weights of the output units do not change any more. Instead of keeping them constant throughout the learning, initializing them with large values and decreasing gradually as in (3.26) and (3.27) generate more reasonable and fast results [21].

$$\eta = \eta_0 \times t^{\alpha} \tag{3.26}$$

$$\sigma = \sigma_0 \times t^{\alpha} \tag{3.27}$$

In the equations, η_0 and σ_0 correspond to initial values of the learning rate and width parameter, respectively. Learning rate is initialized in the range of [0:1], while width parameter can be initialized with a value up to maximum distance between the output units. Besides, t represents the number of updates and α is the amount that the learning rate decreases with [63]. The valid range for α parameter is [-∞:0].

Figure 3.4 represents an example to SOM categorization, when input patterns are in two-dimensional space and 10x10 outputs units in a grid form are used. Input patterns, represented in the left part of the figure, are initialized to have higher density around center. Corresponding output units distribution are represented at the right side of the figure.

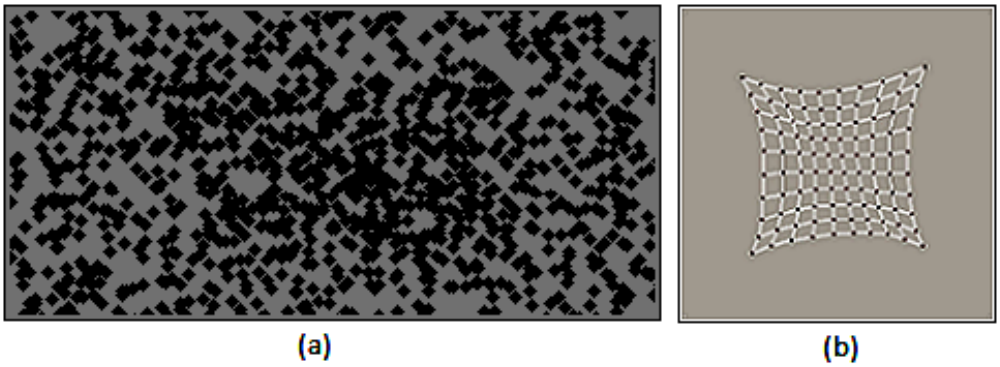

Figure 3.4: SOM categorizaton example

3.4 Recurrent Neural Networks

Recurrent Neural Networks and revised versions of it have been used by the several studies on behavior recognition and learning tasks. Therefore an introductory information about the RNN is presented in this section.

Recurrent Neural Network is a kind of artificial neural network that it has a feedback connections in the network which distinguishes it from the feed-forward neural networks. Thus, a neuron output at time *t* is processed by a neuron in the network at time *t+1* [20][44]. Figure 3.5 represents a sample network for such kind of models.

The feedback loops in the Recurrent Neural Networks adds important learning capabilities to the model. The model has a wide range of use at time series prediction tasks. With the addition of new neurons and by changing the feedback locations, not only one cycle older data, but also past activities can be recorded in the model. Thus, older historical information can be used at the learning, and wider range of problems can be solved by the model.

3.5 Hidden Markov Model

Similar to RNN presented in the previous section, Hidden Markov Model and revised versions of it have been used by the several studies on behavior recognition and learn-

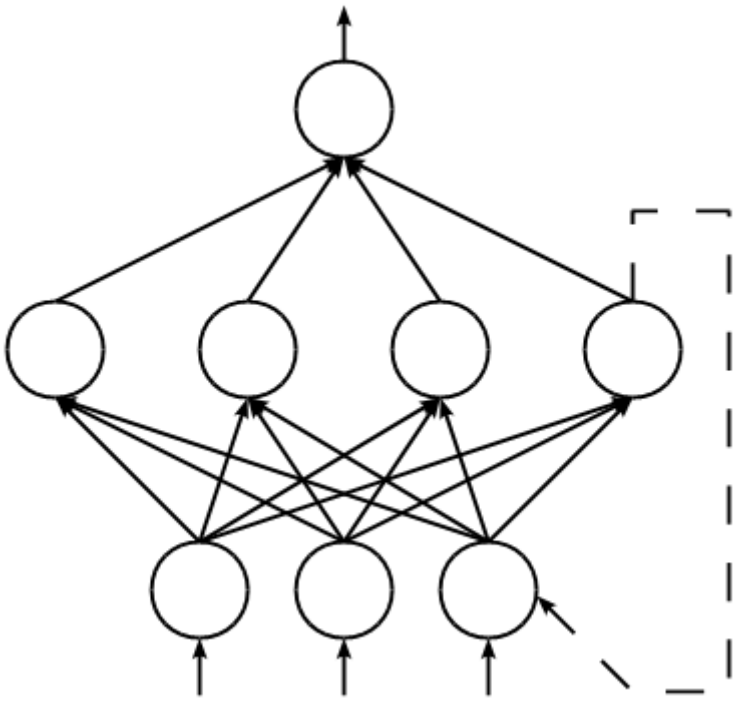

Figure 3.5: A sample Recurrent Neural Network

ing tasks. Therefore an introductory information about the HMM is presented in this section.

Hidden Markov Model is a kind of stochastic finite state machine such that transition between states are done according transition probabilities. Figure 3.6 represents a general schematic of the model. Basically the HMM consists of the following elements [55]:

- Finite number of states.
- Finite number of observation items.
- Stochastic transition matrix that holds a probability of a transition from one state to another. Summation of transition probabilities from any state equals to one.
- Stochastic observation matrix that holds a probability of a seeing an observation item in a state. Summation of observation probabilities for any state equals to one.

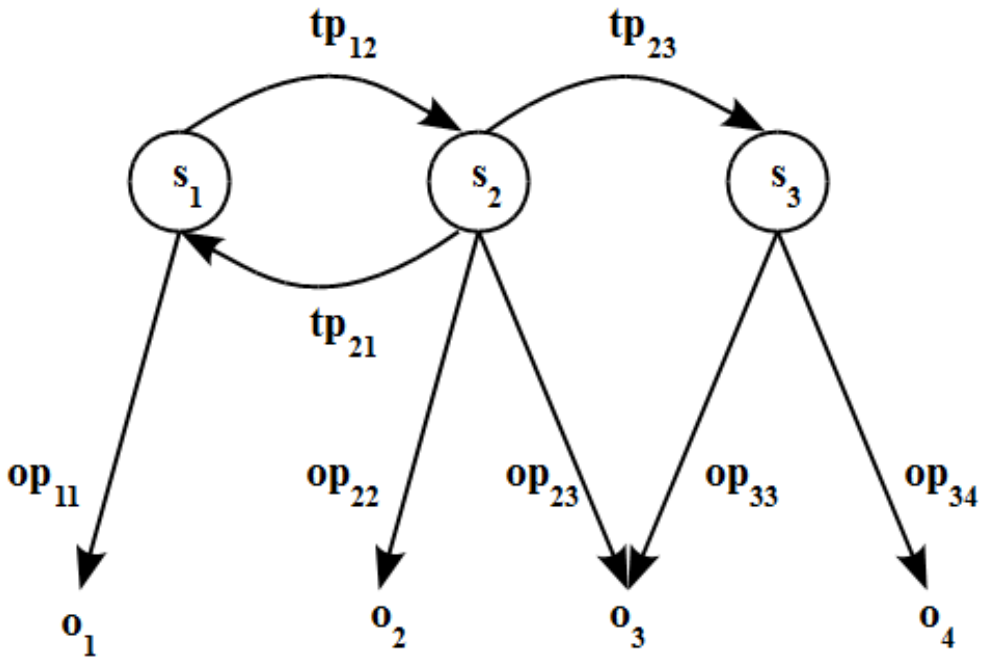

Figure 3.6: General schematic of Hidden Markov Model

- Stochastic initial distribution vector that holds the probability of being at that state at time zero.

CHAPTER 4

CORRELATION BASED ADAPTIVE RESONANCE THEORY

The CobART model [62] is inspired from ART 2. It is developed to enhance the categorization results for the domain having consecutive inputs [61]. The idea of having attentional and orienting subsystems and the relation between them are preserved as in ART 2. Moreover, processing the inputs and updating the categories at the attentional subsystem, and applying the matching functionality at the orienting subsystem are adapted to have better categorization. Correlation waveform analysis and Euclidean distance correlation analysis methods are used as a base for matching process. When the inputs are fed into the network, a category level matching is searched as in ART 2 networks. If an adequate matching is found, the input pattern is mapped with the winner category. If no category is matched by the network, a new category is constructed using input pattern data.

4.1 Structure of CobART Network

Figure 4.1 represents the general structure of the CobART network [63]. The inputs fed into the network are filtered at the w nodes in order to reduce the noise effect using low-pass second-order filters [49]. Low-pass filter is a kind of filter that passes low frequency signals and attenuates high frequency signals. Filters in the form of equation (4.1) are known as second-order[1] filters. Potential category mismatches and

[1] Filter coefficients can be calculated by using algorithms from [12]. Besides, numerical computation tools (e.g. MATLAB [42] - the language and the environment of technical computing) can be used to perform discrete time Chebyshev filter method, which finds similar coefficients. Chebyshev filter is a kind of analog or digital filter that has ripples in the passband [40][41].

creation of unnecessary categories due to noise are eliminated at start with the use of filters. Depending on the characteristics of the input patterns, filtering can be bypassed or any type of low-pass filter can be applied [62].

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) - b_1y(n-1) - b_2y(n-2) \tag{4.1}$$

The matching scores between the processed input data and the generated categories are calculated at r nodes [61]. Correlation Waveform Analysis (see Section 3.1.2) and Euclidean Distance Method (see Section 3.1.1) are used for the category matching process, as given in (4.2). Because the distance and shape of change are both important for the matching, we combine these two method.

$$r_k = CWA(w, z_k) \times b + EDM(w, z_k) \times (1 - b) \tag{4.2}$$

In equation (4.2), r_k corresponds to the matching score between the processed input data w and k^{th} category data z_k [63]. Because CWA and EDM calculates correlation between two vectors (one dimension of input data and corresponding dimension of the category data in the equation), not between two values, index i in the figure is not used in the equation. Matching score is calculated based on the weighted average of CWA and EDM, and the weights of these methods are arranged by the b parameter. Parameter b is set in the range of [0:1].

Once a matching score for each category is calculated, the highest r node is selected as the matching score and is compared with a vigilance parameter ρ [62]. The vigilance parameter determines the quality of learned categories. The more the value of ρ is, the more vigilant the system becomes, generating finer distinction. If the matching score is higher than ρ, the winner category is determined. Then, the winner category data are updated according to (4.3).

$$z_{Ki} = z_{Ki} + \eta \times (w_i - z_{Ki}) \tag{4.3}$$

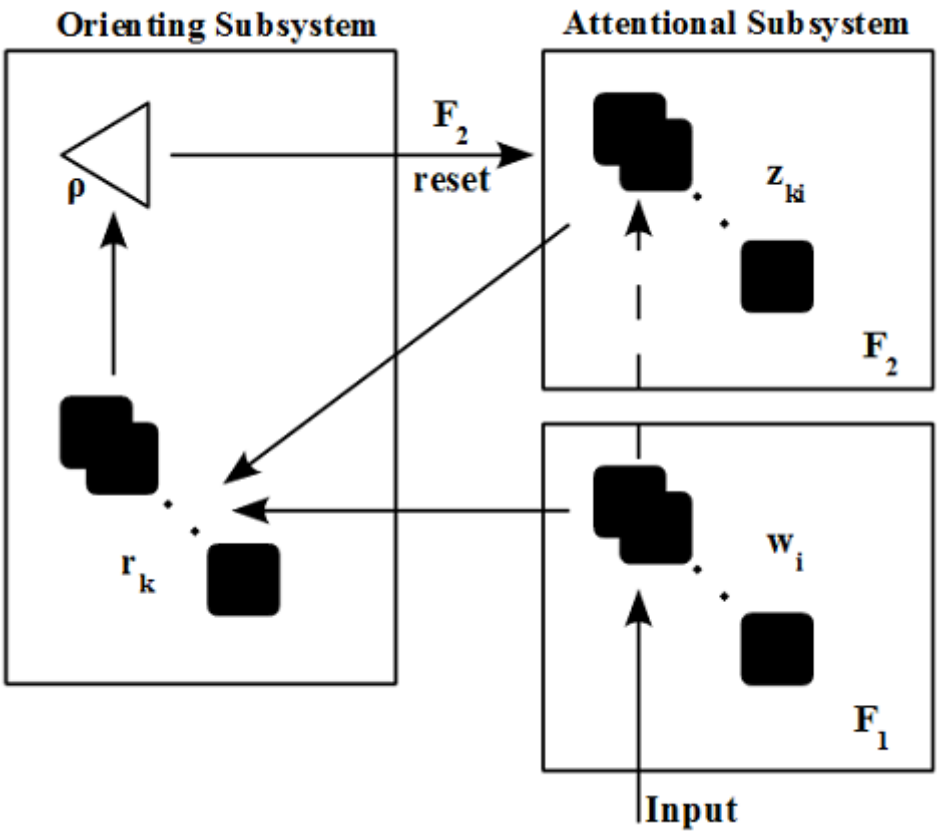

Figure 4.1: Structure of CobART network [62]

In equation (4.3), the winner category data z_K are changed in accordance with the learning rate η and with respect to the deviation from the processed input data w_K [62]. Learning rates of the learned categories are decreased according to (4.4) over time in order to protect learned categories from endless changes. The learning rate calculation is inspired from Kohonen networks [33] (see Section 3.3). It starts from an initial learning rate value d, which is in the range of [0:1], and goes down to zero.

$$\eta = t^{\alpha} \times d \tag{4.4}$$

In equation (4.4), t represents the number of updates over the each category and α is the amount that the learning rate decreases with [63]. The valid range for α parameter is [-∞:0].

If the matching score is smaller than ρ, category matching fails and a new category is created immediately [62]. This new category data is initialized using w node values in order to enhance valid initial data.

4.2 CobART Evaluation

CobART network categorization adequacy has been compared with ART 2 network categorization using 50 types of analog input pattern categorization task, which is presented in [4]. ART 2 categorization results can be seen in Figure 4.2 [62], where ART 2 categorized these analog input patterns into 34 categories. The resultant categorization is reached after a single presentation of each input pattern. When successive presentations of the same inputs are tested, no change at the resultant categories is observed. In [4], same analog input patterns are categorized into 20 categories at another experiment by using smaller vigilance parameter. But, the results of that experiment is not selected for comparison because of the existence of less reasonable categorization.

CobART network training is done by feeding these analog input patterns into the network five of times in random sequences in order to get rid of the pattern sequence

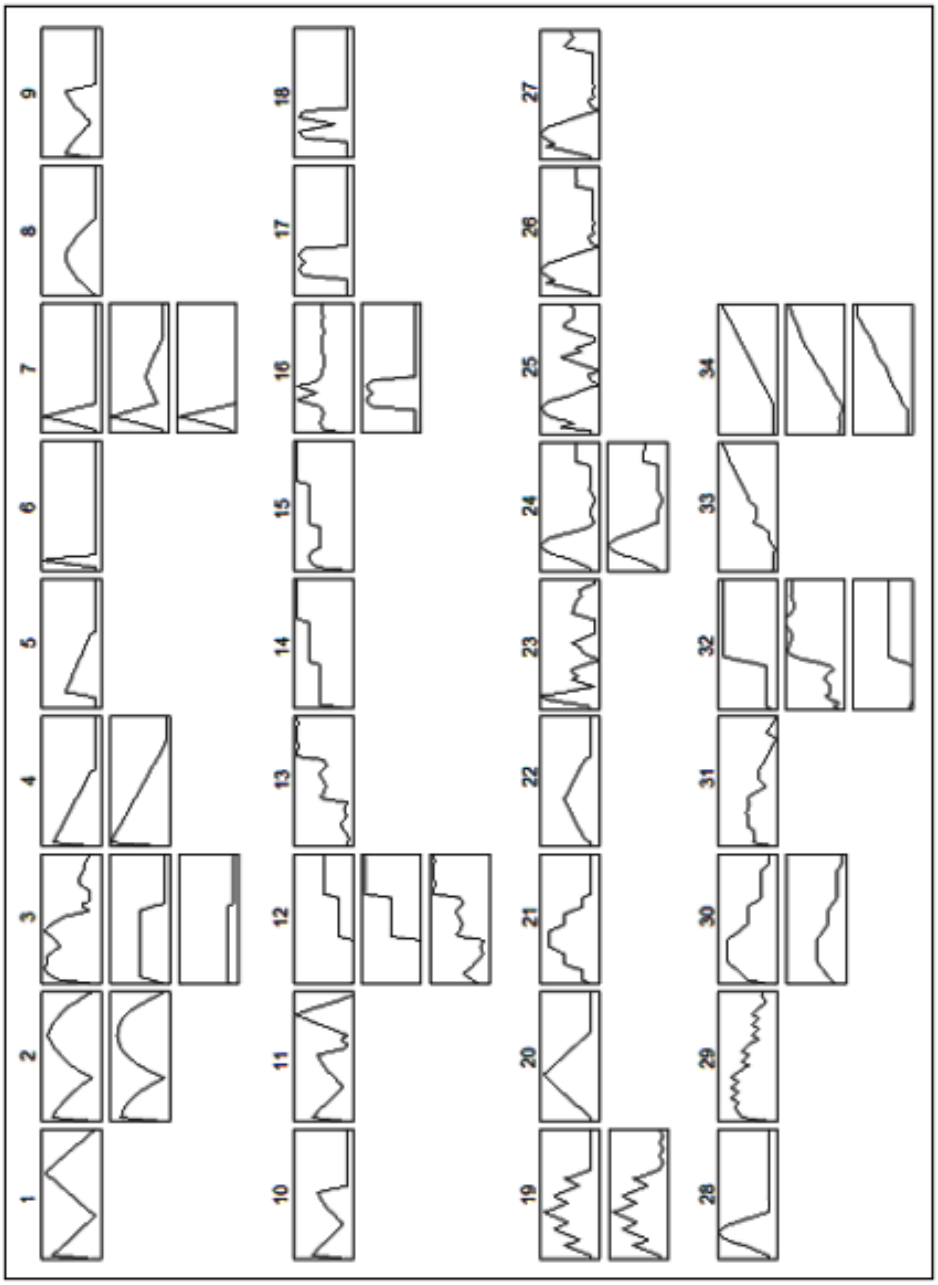

Figure 4.2: Categorization using ART 2 [4]

effect over the categorization process. CobART categorization results for the same input patterns can be seen in Figure 4.3. This categorization is obtained when vigilance parameter ρ is 0.9, b is 0.75, d is 0.2, α is -1.5 [62]. Because input patterns are not instant runtime data, but the prepared analog patterns, filtering at w nodes is bypassed.

Comparisons of the categories from Figure 4.2 and Figure 4.3 show that CobART generates more reasonable categorization than ART 2 [62]. CobART categorized 50 types of analog input patterns into 24 categories, while ART 2 categorized them into 34 categories having 23 of the categories are created for a single analog input pattern. Some of the ART 2 categories having only one input pattern may have been combined with other categories. For example;

- ART 2 categories 1 and 2 are grouped into category 1 by CobART.
- Input patterns in ART 2 categories 9 and 10 are mapped by the same CobART category, which is category 9.

Moreover, some ART 2 categories are not self-coherent that some of the input patterns grouped into these categories may have been mapped by different categories. For example;

- ART 2 category 3 is divided into categories 2 and 3 by CobART.
- Last pattern of ART 2 category 12 has less coherency with the other patterns of the same category when compared with the pattern in category 13. CobART grouped them within the same category.

These are only a few samples of similar cases. On the other hand, the main deficiency of CobART is mapping of the last pattern in category 24 whereas it should be categorized as category 11 or as a new one. In fact, it is mapped with category 11 at some runs and these 3 patterns from category 11 are sometimes grouped with the patterns in category 24, even when no parameter is changed because of the effect of random-

ization at training phase. It is tested that, increasing ρ solves this problem, but this solution causes some other categories to be divided into subcategories as expected.

The effects of the CobART network parameters are tested with different experiments. When vigilance parameter is increased, number of categories is approaching to the number of patterns [62]. When it is decreased, patterns are grouped into fewer categories. Experiments over b parameter showed that increasing the effect of CWA ($0.5 < b < 1$) generates better categorization. However, keeping effect of EDM helps to construct new reasonable categories for the patterns having similar shapes but different amplitudes. For example, when b is set to 1, patterns grouped into the categories 22 and 23 are mapped with the same category. In order to test filtering effect at w nodes, filtering is activated and tested by changing cutoff frequency[2]. As a result, it caused some categories to be combined. For example, patterns mapped by the categories 16 and 17 are grouped within the same category when filtering is tightened.

[2] The highest frequency that is passed from the low-pass filter without attenuation is known as cutoff frequency.

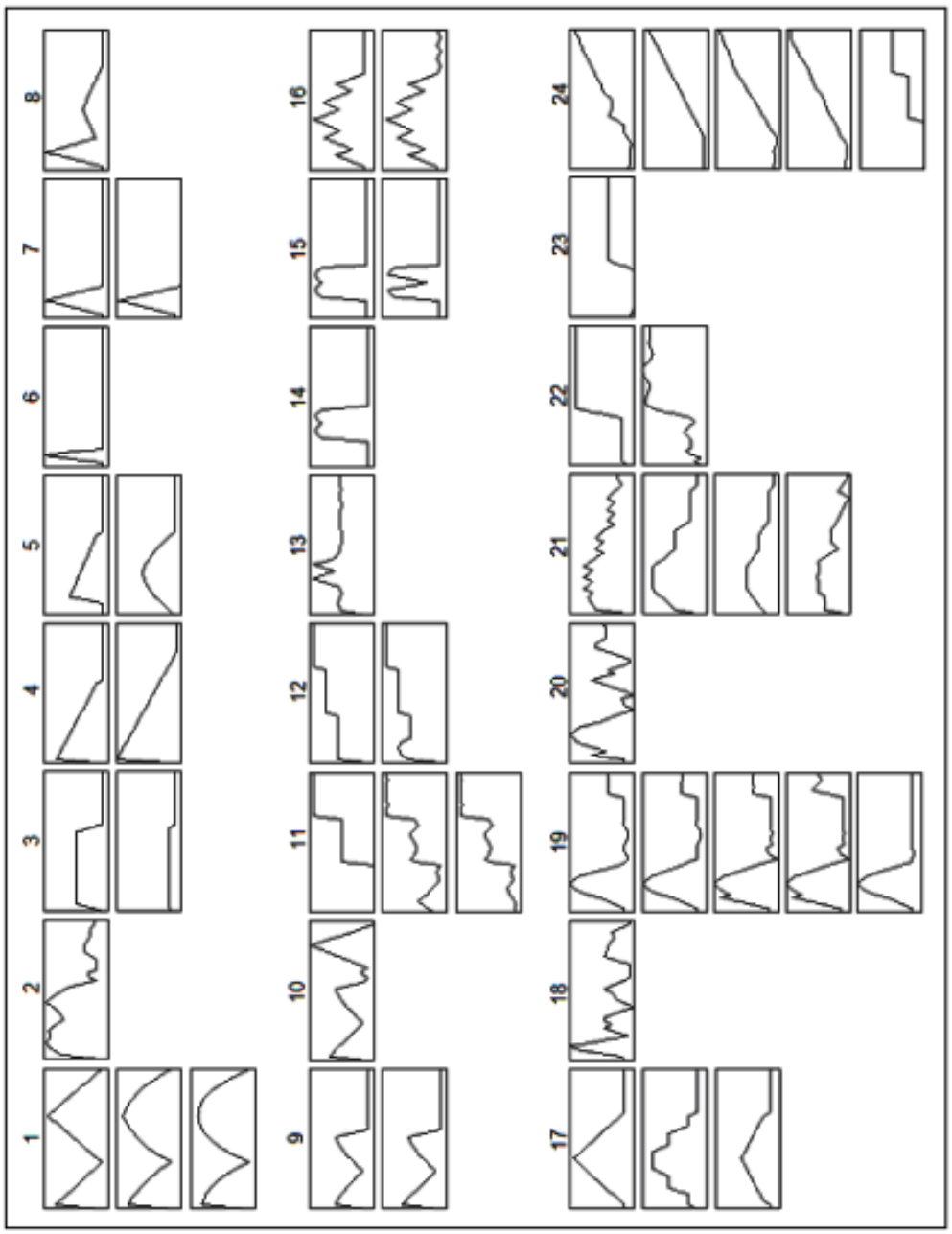

Figure 4.3: Categorization using CobART [62]

CHAPTER 5

BEHAVIOR CATEGORIZATION

Developing an adequate behavior categorization model to find out the primitive behaviors is the main purpose of this study. A generic behavior model can use these primitive behaviors for behavior recognition and learning tasks, and when adapting to new environments. Therefore, this model is desired to be unsupervised and self-organizing. It should expand itself for new behaviors, and it should not forget the learned ones.

ART 2 was a good candidate to be used in such a model. Hence, the studies on the behavior categorization was started by implementing ART 2 network. A wide range of tests on the ART 2 parameters were not resulted in reasonable categorization; and priori SOM categorization was combined with the ART 2. Because this new architecture did not enhance the behavior categorization results, CobART was developed and used for the behavior categorization. CobART generated reasonable categories for the behaviors being tested. It can differentiate different behaviors, and alternative forms of the same behaviors. To learn the relations between these categories, CobART networks are combined hierarchically.

These models, test environment and the evaluation results are presented and discussed in detail, in the following sections.

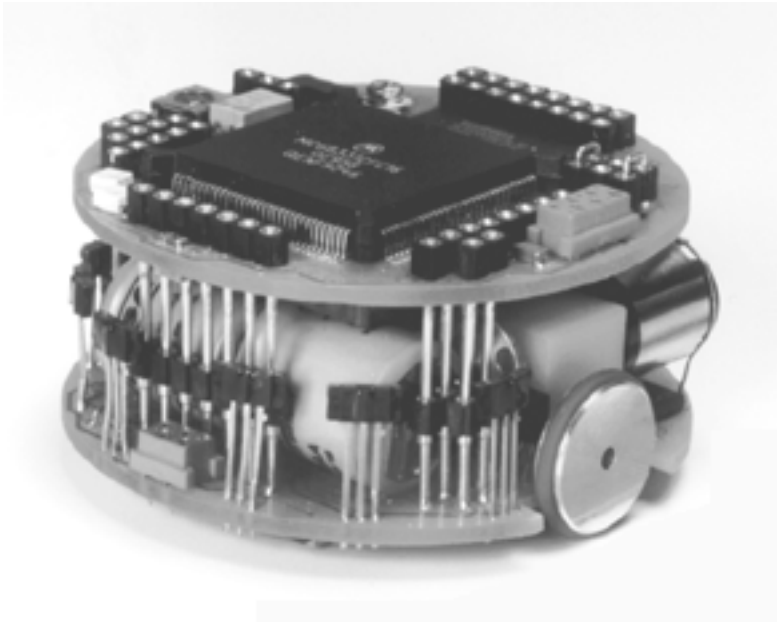

Figure 5.1: Khepera [35]

5.1 Test Environment

The models are tested on behaviors such as *pushing*, *approaching*, and *striking* in order to find out the capabilities of the models [63]. *Webots* simulator was programmed for each behavior to collect *Khepera*'s own sensory inputs as a training and test data. Then, multiple runs were performed and motion data was logged for each run. To collect various data, the robot was made to move towards a target object from various distances by having varying velocities.

5.1.1 Khepera

Khepera was designed as a research and training tool by LAMI (Microprocessor Systems Laboratory) group of EPFL university [29] in 1996. Initial version of the Khepera robot is presented in Figure 5.1.

Specifications of the Khepera are as follows [35]:

- It runs on Motorola 68331 16 MHz processor

- 512 KB EEPROM
- 256 KB memory
- Two DC motors with incremental encoders. Incremental encoders generates 600 pulses per wheel revolution
- Eight infrared proximity and light sensors
- K-Extension bus provides expansion with external modules such as gripper
- Serial port can be used to communicate with development environment
- It has a 55 mm diameter, 30 mm height and 70 g weight

5.1.2 Webots Simulator

Webots Manual defines Webots as a "three-dimensional mobile robot simulator that was originally developed as a research tool for investigating various control algorithms in mobile robotics" [38].

Starting from the third version of the simulator (the version that is used in this study), "any robot with two-wheel differential steering can be modeled and simulated. Predefined objects like shapes, sensors and axles allow users to create and run their own simulated robot" [38].

Webots is a license based application, and it can run on Linux (Redhat 7.2 and next generations) and Windows (Windows 95 and next generations) operating systems. Webots for other operating systems MacOS, Solaris, Linux PPC, Irix can be found upon request [38].

Webots 3.0 contains example packages for the following robots:

- Khepera
- Koala
- Alice

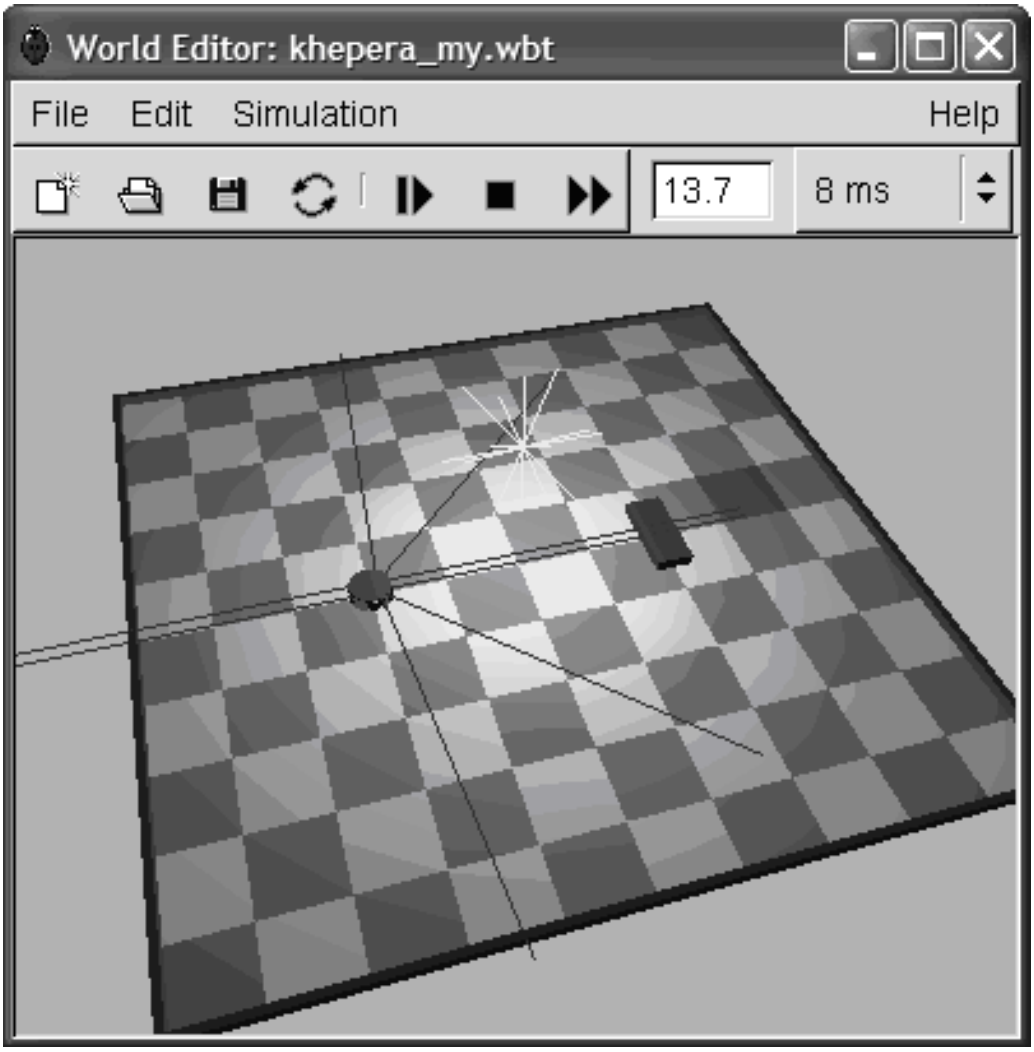

Figure 5.2: Webots virtual world window

- Pioneer 2

Webots contains three types of components to handle simulation task:

- World: Is a kind of configuration file that determines the features of the environment and the robot. Webots has an editor that configures and displays world file content. A sample world window is presented in Figure 5.2.
- Controller: Is a kind of application that controls a robot by reading sensors, processing them and writing commands to actuators. Controller applications are generally written in C, C++ or Java languages.

- Supervisor: Is a kind of application that controls and tracks world environment in runtime. It can change the positions of the objects and the robots contained in the world. Supervisor applications are generally written in C or C++ languages.

5.1.3 Collecting Training and Test Data

We used Webots 3.0 simulator Windows version to collect the training and test data. Although lots of robot alternatives exist for this simulator, Khepera is selected because of its simplicity, and in order to have a chance of trying future experiments on real Khepera robot at hand. The robot with test environment is represented in Figure 5.2.

Infrared sensors of the Khepera are represented in the figure by the black lines connected to the robot. The Infrared sensors located at the backs and sides are not used for data collection. Distance to the object is calculated using two front infrared sensors, and position, velocity and acceleration values are calculated using encoder inputs. Maximum distance that can be measured by the infrared sensor is set to 50 cm within the world file in order to have wider range of testing area. Besides, velocity unit is decreased to 0.01 rad/s in order to work with smaller velocities.

A controller application is written for each behavior to collect *Khepera*'s own sensory inputs. It is a simple program that, basically reads infrared sensors and encoder values, and generates motion commands to the actuators to control the wheel motion. Besides, it logs position, velocity, acceleration and distance values into an output file. To collect various data, the robot was made to move towards a target object (black box in the figure) from various distances by having varying velocities. For *approaching* behavior, controller application generates constant velocity until the robot approaches to the target object, and then slows down gradually to zero. For *striking* behavior, it generates constant velocity until touching to the object. When it is touched, it generates slowing down velocity command at the reverse direction. For *pushing* behavior, controller application generates constant velocity until touching to the object. After touched, it keeps generating motion for the robot. But, the touched object should start moving to simulate pushing behavior. A supervisor application is written for this pur-

pose. Supervisor application is started moving the object when a distance between the Khepera and the object is less then some predefined value. Velocity of the object is changed for each test in order to simulate pushing objects having different weights.

5.2 Behavior Categorization Using ART 2

ART 2 network is implemented according to an approach based on [4]. It is important to determine the input data and the parameters of the network. Velocity, acceleration, and object distance data are selected as inputs. They are fed into the network not as an instant data but as a consecutive input data[1]. First 5, then 10, and 20 consecutive data sets are fed into the network but no considerable improvement is obtained [61]. Categorization gave better results only after acceleration data are excluded from the input data, because of high fluctuations in the acceleration data. More satisfactory results are reached when touched status is added as an input to the network.

Equation (5.1) represents categorization process at time t with c_t symbol. In the equation, N corresponds to consecutive input number[2]. Besides, vel_t, $dist_t$ and $touched_t$ correspond to the velocity, the distance and the touched status data at time t respectively. During the experiments, test instances are labeled as test numbers from 1 up to 2400 [63]. When analyzing the results, these test numbers will be referred to point the corresponding behavior data in the figures.

$$\begin{aligned} c_t \;=\; & ART\ 2(vel_t, .., vel_{t-N+1}, \\ & dist_t, .., dist_{t-N+1}, \\ & touched_t, .., touched_{t-N+1}) \end{aligned} \tag{5.1}$$

The categorization results are represented in Figure 5.3 and Figure 5.4 [61]. These results are reached when the ART 2 parameters a and b are set to 1.2, vigilance parameter ρ is set to 0.96, and 10 consecutive data is used. The velocity and the

[1] Input data is a two-dimensional vector. First row holds newest velocity, acceleration and distance data, while other rows hold preceding values respectively.

[2] The period of the behavior being categorized.

distance data are scaled down to [-1:1] and [0:1] ranges respectively before training and testing, in order to set the effect of each data similar.

The categorization results of 10 behavior data segments corresponding to 1950[3] test data are presented in the figures [63]. These segments are selected from different parts of the long training data in order to simplify the analysis process and to increase the understandability. The training data contains around 18000 samples, each having robot velocity, robot acceleration, robot position, object velocity, and distance values. Touched status is calculated from the distance data. The training is performed by five iteration over the training data.

Data segments in the figure are separated by thick vertical lines and labeled as -d_1- through -d_{10}- at the top of the segments [63]. In the figure, left axis indicates the inputs' value for the velocity data, the distance data and the touched status. Besides, right axis indicates the category identifiers, and the horizontal axis indicates the test numbers. Thin black line and light gray clipped line are used for the velocity and the distance graphs respectively, while dark gray triangle is used for the touched status, and black circles labeled by numbers are used for the categories.

According to human point of view, the behavior segments in Figure 5.3 and Figure 5.4 correspond to the following behaviors [61]:

- d_1, d_2, d_4 and d_6 segments are approaching behaviors having velocity with decreasing distance.
- d_3, d_7 and d_8 are striking behaviors having reverse velocity at some point with increasing distance.
- d_5, d_9 and d_{10} are pushing behaviors having velocity with zero distance.

It should be pointed out that, pushing behavior data segments in the graphs generally have approaching behaviors at the beginning in order to test how the models respond to change of behavior. Besides, striking behaviors have the same approaching data at the beginning because of the nature of the striking behavior.

[3] Test numbers in the figures reaches up to 2400 because of spaces between the data segments.

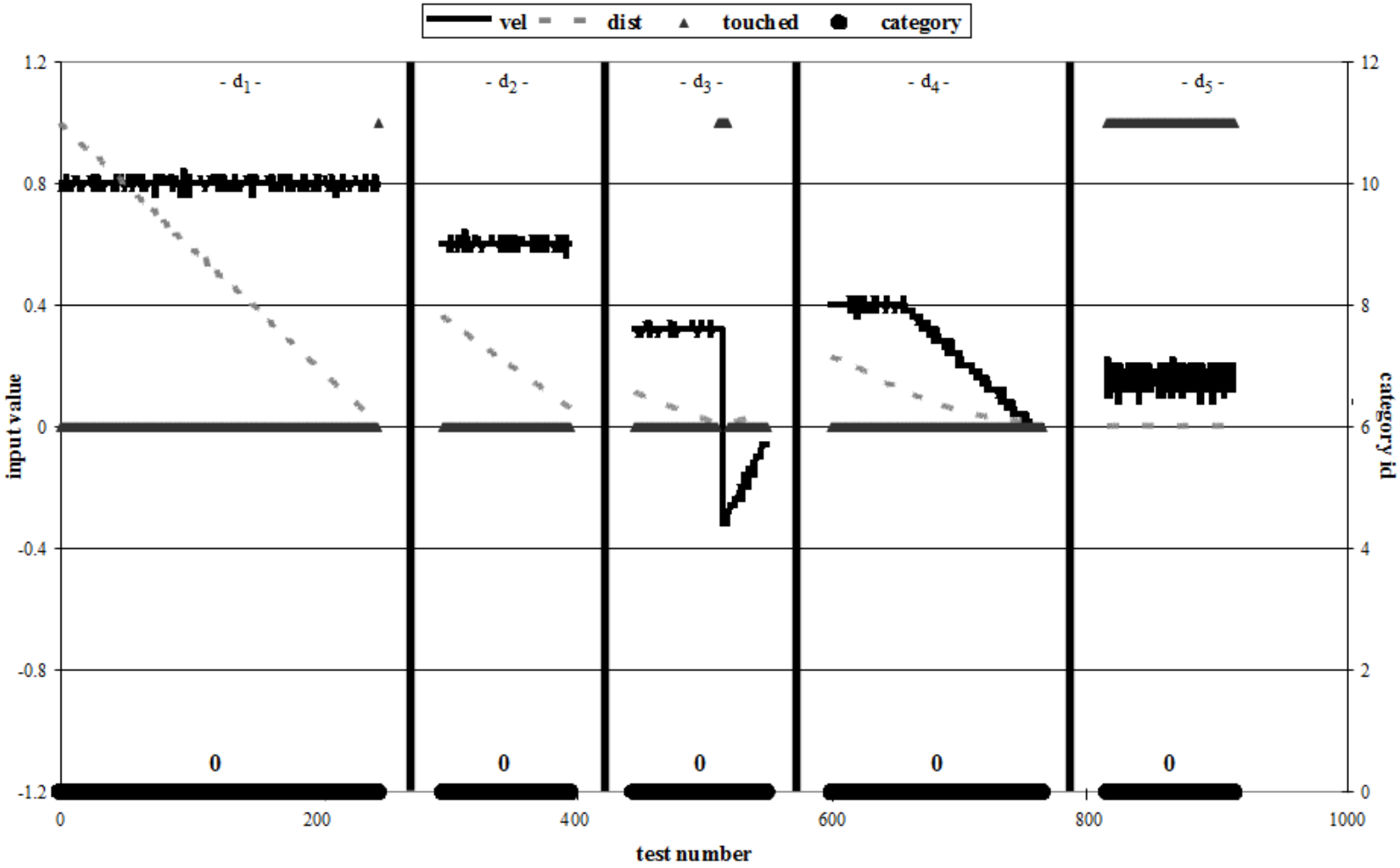

Figure 5.3: Behavior categorization using ART 2 - 1

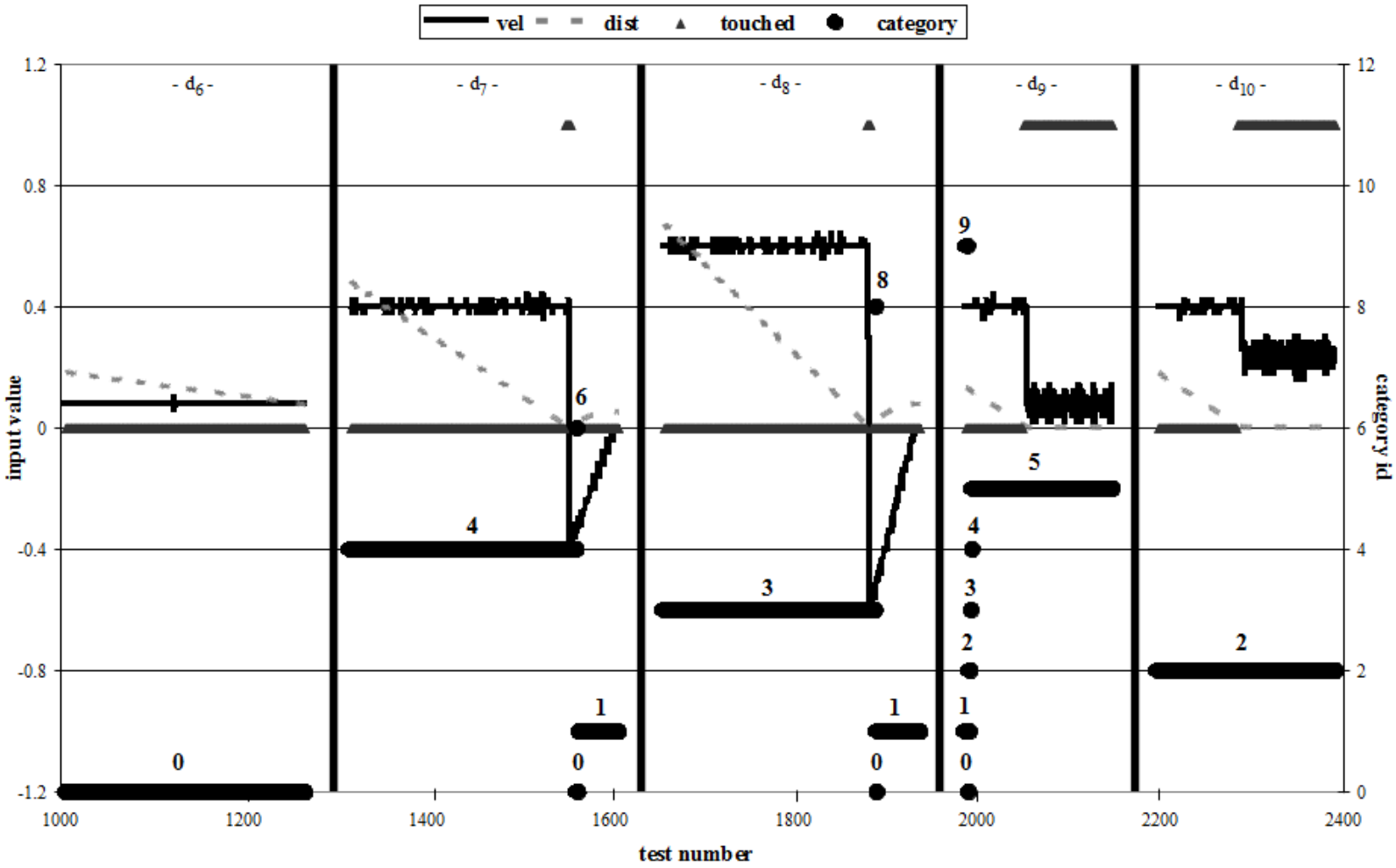

Figure 5.4: Behavior categorization using ART 2 - 2

Table 5.1: ART 2 categories generated for corresponding behaviors

data segments	approaching	pushing	striking
d_1	0	-	-
d_2	0	-	-
d_3 (before 530)	0	-	-
d_3 (after 530)	-	-	0
d_4	0	-	-
d_5	-	0	-
d_6	0	-	-
d_7 (before 1580)	4	-	-
d_7 (after 1580)	-	-	1
d_8 (before 1900)	3	-	-
d_8 (after 1900)	-	-	1
d_9 (before 2120)	1,5	-	-
d_9 (after 2120)	-	5	-
d_{10} (before 2320)	2	-	-
d_{10} (after 2320)	-	2	-

Table 5.1 shows the behaviors performed at each data segment, and lists the corresponding categories generated by ART 2. Data segments for pushing and striking behaviors are explained in two rows if they have approaching behavior preceding it. Test numbers corresponding to the duration of going from one behavior to another are written next to the data segment identifiers.

Although some improvements are seen after touched status is added, better categorization is needed for successful behavior recognition [61]. It can be seen from the Figure 5.3, Figure 5.4 and Table 5.1 that, behaviors are not stabilized even after long-time learning. Similar behavior data are categorized by different categories, and unrelated data are grouped within same categories. For example,

- Similar approaching behaviors at data segments d_2 and d_8 are classified with different categories.
- Different approaching behaviors at data segments d_1, d_3 and d_6 are mapped by same category 0.
- Pushing behaviors at d_5, d_9 and d_{10} are classified with different categories 0, 5, and 2, although these categories are also mapped by approaching behaviors.

These deficiencies arise from the fact that continuous update of categories loses all/some information about old categories. Besides, ART 2 focuses heavily on the significant inputs values when selecting winner category, while correct behavior recognition needs input derivation to be taken into account.

5.3 Behavior Categorization Using SOM and ART 2

When SOM network is combined with ART 2 network, inputs to the model are fed into the SOM network first. Then, categorized SOM network outputs are used as an inputs to the ART 2 network. The velocity and the distance data are processed by SOM network after they are scaled down to [-1:1] and [0:1] ranges respectively. On the other hand, because touched status is already in the discrete domain: {0,1}, it not processed by SOM.

$$\begin{aligned}(vel_t^s, dist_t^s) &= SOM(vel_t, dist_t) \\ c_t &= ART\ 2(vel_t^s, .., vel_{t-N+1}^s, \\ &\qquad dist_t^s, .., dist_{t-N+1}^s, \\ &\qquad touched_t, .., touched_{t-N+1}) \end{aligned} \tag{5.2}$$

Equation (5.2) represents categorization process at time t with c_t symbol. In the equation, vel_t^s and $dist_t^s$ correspond to categorized velocity and distance data by SOM, while N corresponds to consecutive input number.

SOM network is implemented according to an approach based on [21]. Two input neurons and 36 output neurons are used, and output neurons are kept in two-dimensional vector having the length of six for each dimension. Output neuron weights are initialized randomly within the range of [0.2:0.8]. Weight update process is repeated as many times as pre-defined [61]. In our experiments, output neuron distribution couldn't become stable until 500 iterations, promising results are obtained after 1000 iteration. At the beginning of each iteration, learning rate η_0 is set to 0.5

Table 5.2: SOM and ART 2 categories generated for corresponding behaviors

data segments	approaching	pushing	striking
d_1	0	-	-
d_2	0	-	-
d_3 (before 530)	0	-	-
d_3 (after 530)	-	-	1
d_4	1	-	-
d_5	-	1	-
d_6	1	-	-
d_7 (before 1580)	7	-	-
d_7 (after 1580)	-	-	8
d_8 (before 1900)	4	-	-
d_8 (after 1900)	-	-	10
d_9 (before 2120)	2	-	-
d_9 (after 2120)	-	2	-
d_{10} (before 2320)	13	-	-
d_{10} (after 2320)	-	13	-

and width parameter σ_0[4] is set to 25. These values are decreased at the rate of α which is set to -0.5.

Figure 5.5 and Figure 5.6 show behavior categorization results by SOM and ART 2 [61], when the ART 2 values a and b are set to 1.2, vigilance parameter ρ is set to 0.96, and 10 consecutive data is used. Priori SOM categorization effect on the input data can be understood when velocity and distance graphs are compared with the graphs in Figure 5.3 and Figure 5.4. With SOM categorization, velocity and distance graphs are discretized.

Table 5.2 shows the behaviors performed at each data segment, and lists the corresponding categories generated by SOM and ART 2. It can be seen from Figure 5.5, Figure 5.6 and Table 5.2 that, using priori SOM categorization didn't increase ART 2 behavior categorization performance. For example,

- Going back after striking, approaching and pushing behaviors at data segments d_3, d_4 and d_5 are classified with same category 1.

[4] Determines neighboring neurons around the winner neuron.

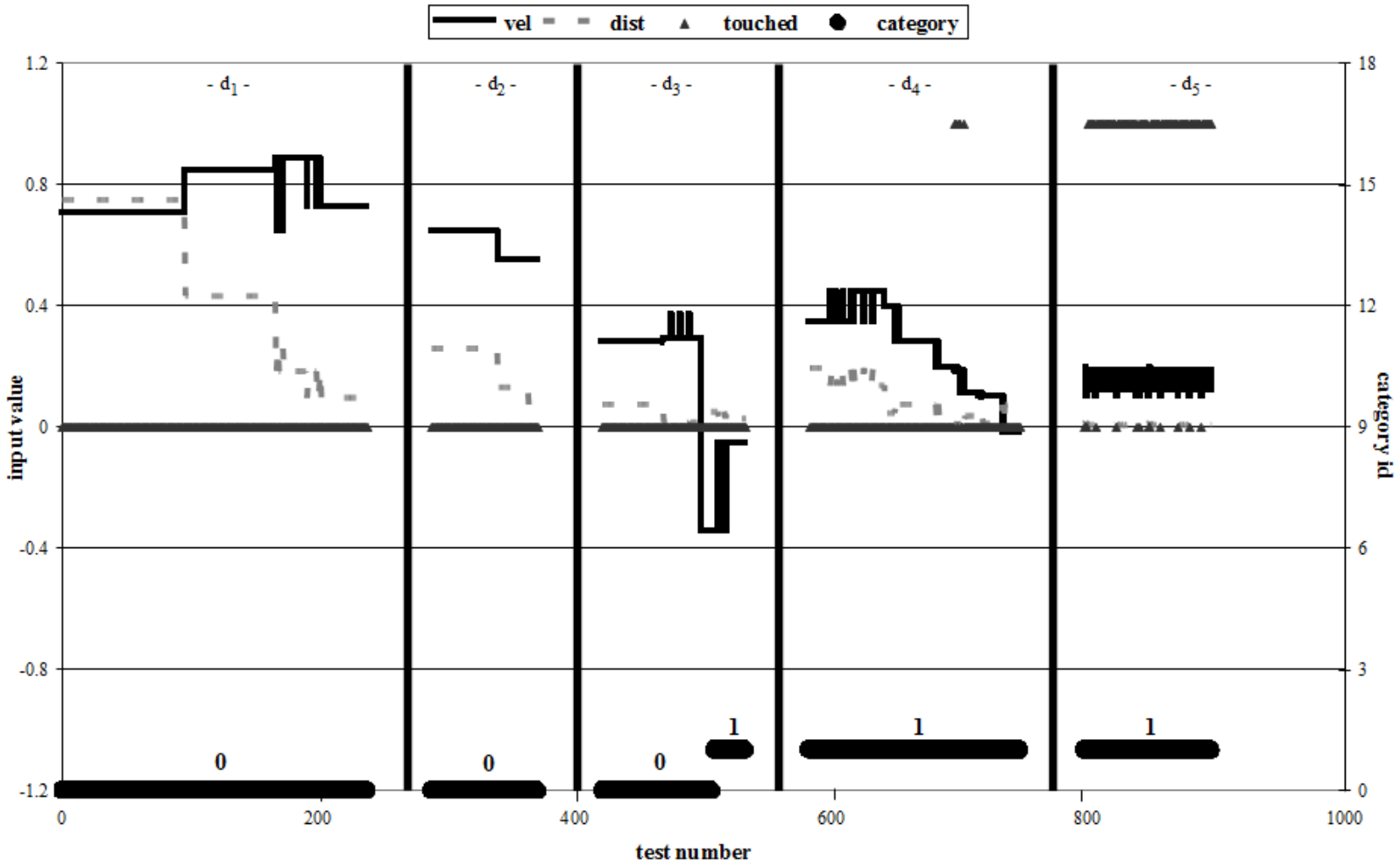

Figure 5.5: Behavior categorization using SOM and ART 2 - 1

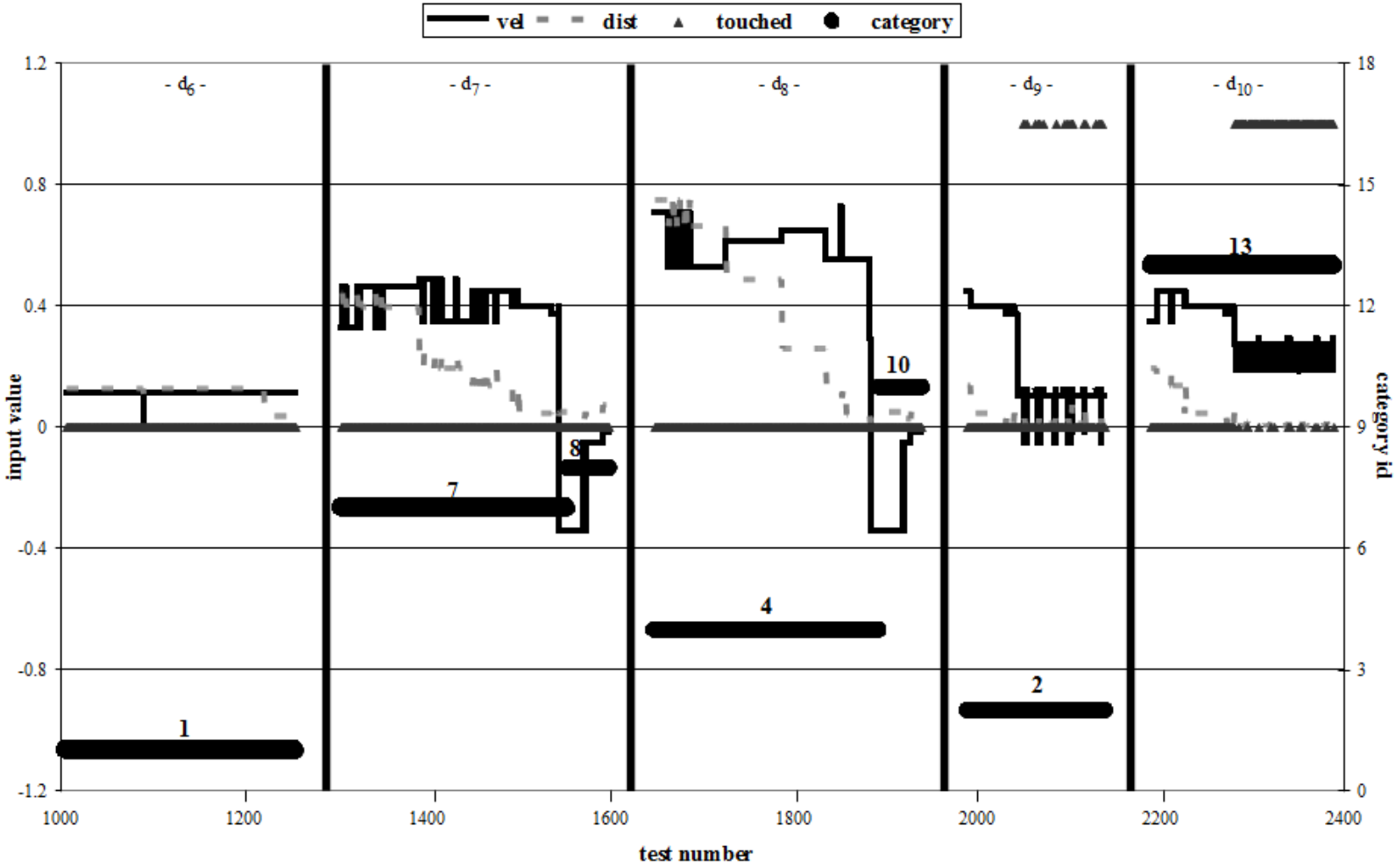

Figure 5.6: Behavior categorization using SOM and ART 2 - 2

- All going back after striking behaviors at d_3, d_7 and d_8 are classified with different categories 1, 8 and 10, respectively.

- Moreover, the same type of pushing behaviors at d_9 and d_{10} are classified with different categories 2 and 13.

5.4 Behavior Categorization Using CobART

CobART network is slightly modified for behavior categorization. The modified network structure is represented in Figure 5.7. The inputs fed into the network are buffered for a pre-defined period of time to have consecutive input data (depicted by index *j*) [63]. Besides, small percentages of active category (if exists) data are summed with the input data at *p* nodes to keep the continuity of behavior data. In the figure, *p* nodes corresponds to the processed input data. Their values are computed by equation (5.3).

$$p_{ij} = w_{ij} \times a + z_{kij} \times (1 - a) \tag{5.3}$$

In the equation, p_{ij} represents the processed input data for the i^{th} dimension of the input vector and j^{th} consecutive input value [63]. Besides, w_{ij} represents the filtered value of the same input while z_{kij} represents corresponding value of the k^{th} category data. Parameter a determines the weight of the input data and the active category. Parameter a is set in the range of [0:1].

Correlation Waveform Analysis method used in the original CobART network is replaced with Derivation Correspondence Method (see Section 3.1.3). DCM is selected in order to enhance behavior categorization by adding derivation similarity check to the matching score calculation.

$$r_k = \frac{1}{I} \times \sum_{i=1}^{I} (DCM(p_i, z_{ki}) \times b + EDM(p_i, z_{ki}) \times (1 - b)) \tag{5.4}$$

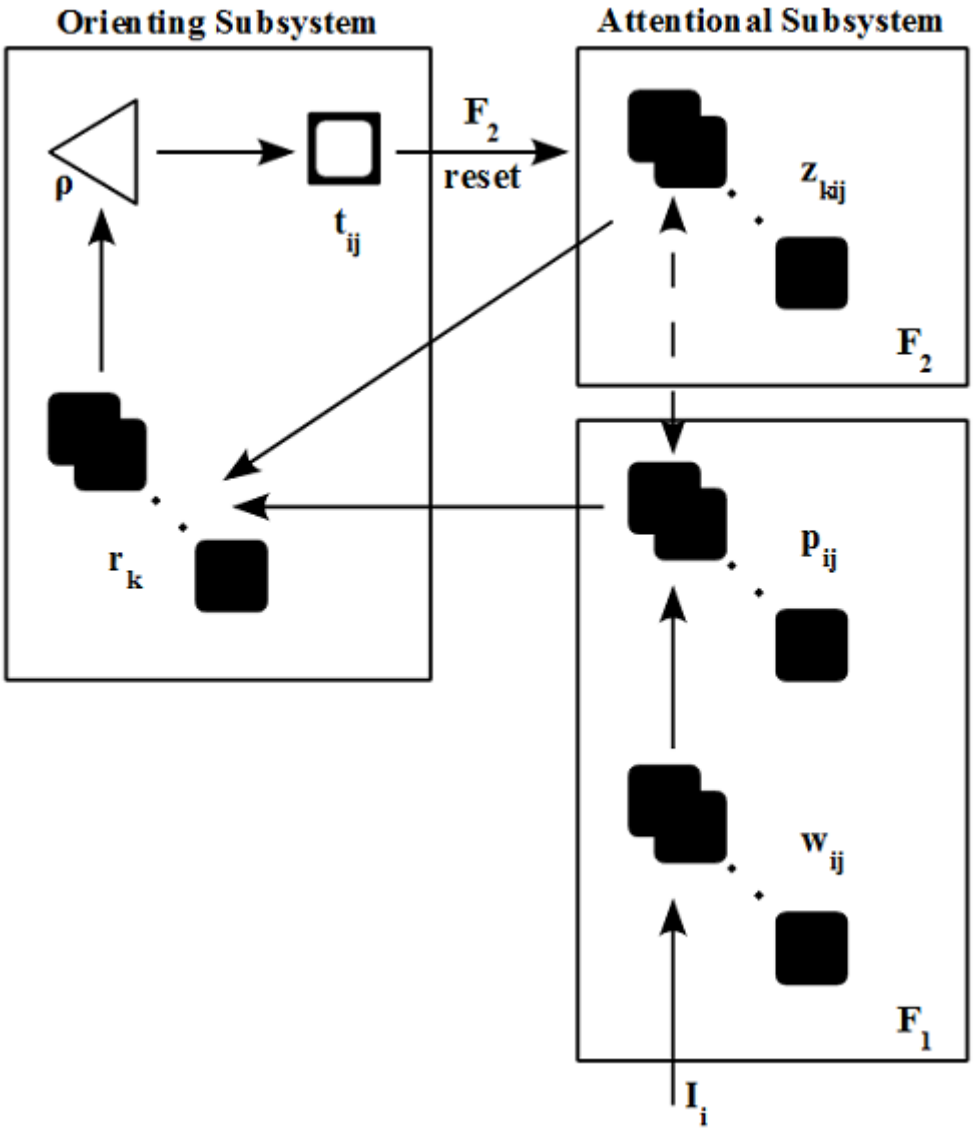

Figure 5.7: CobART network for behavior categorization [61]

In equation (5.4), r_k corresponds to the matching score between the processed input data p_i and k^{th} category data z_{ki} while I represents the input vector length. The calculated is made based on the weighted average of DCM and EDM. The weights of these methods are arranged by the b parameter. Matching score is calculated for each dimension of the behavior data separately, and their summation is divided by the input vector length.

The last modification to the CobART network is related with the time to create a new category. When category matching fails, a new category is not created immediately, but the last winner category stays active for a few steps. New category is created when successive matchings has failed [63]. Because, noise may cause unnecessary category creation, and categories can match to the behaviors from different starting points [61]. While temporary winner is active, network is prepared to a plausible new category at the same time using a t node. This node is updated with p node values in order to enhance a valid initial data for a new candidate category.

5.4.1 CobART Behavior Categorization Evaluation

The tests are started with experiments on different types of correlation methods. When CWA is used, satisfactory categorization can not be generated. EDM was tested but failed at discriminating between pushing and approaching behaviors at some points. The reason is that EDM method focuses on how far two vectors are to each other. New type of correlation method is needed which also takes derivation as a parameter. When DCM was combined with EDM, more reasonable categorization is reached. The tests over parameter b showed that increasing the effect of DCM ($0.5 < b < 1$) generates more reasonable categorization. Parameter a also affects the quality of the categorization. More reasonable categorization is reached when it is close to one ($0 << a < 1$). When a is set to 1, unnecessary jumps between the categories are observed, and stable categorization cannot be achieved for stable behavior data.

Figure 5.8 and Figure 5.9 represent the CobART behavior categorization results when velocity and distance data are given as inputs to the network [63]. Equation (5.5)

represents the categorization at time t, which is denoted by c_t. In the equation, N corresponds to the consecutive input number, vel_t and $dist_t$ correspond to the velocity and the distance at time t respectively. In fact, only vel_t and $dist_t$ are fed into the CobART network at time t. They are combined with the last *N-1* inputs within the network before categorization. In the experiment, 10 consecutive data are gathered at each w nodes, and tight filtering is enabled. Cutoff frequencies for filtering of the distance and the velocity data are set to 2.5 Hz and 1 Hz respectively, while sampling frequency[5] is set to 50 Hz. Vigilance parameter ρ is set to 0.8, α is -1.5, d is 0.2, a is 0.8, b is 0.7, and N is 10.

$$\begin{aligned} c_t &= CobART(vel_t, .., vel_{t-N+1}, \\ &\qquad dist_t, .., dist_{t-N+1}) \end{aligned} \tag{5.5}$$

A detailed view of the categorization can be seen in Figure 5.10 [63]. It shows the categorization results from the data segment d_3 for the test numbers 535 through 560. In the figure, left and right axes are re-scaled and the markers are used for the velocity and the distance data in order to express inputs and the generated categories more clearly. The categorization process is illustrated better by the following example. Equation (5.6) represents the data fed into the categorization process when category 14 is firstly generated for the test number 555.

$$\begin{aligned} c_{555} &= CobART(vel_{555}, .., vel_{546}, \\ &\qquad dist_{555}, .., dist_{546}) \end{aligned} \tag{5.6}$$

Table 5.3 shows the behaviors performed at each data segment, and lists the corresponding categories generated by CobART [63]. It can be observed from Figure 5.8, Figure 5.9 and Table 5.3 that, only similar behaviors are mapped to the same categories. For instance,

[5] Sampling frequency corresponds to a number of data sampled in a second.

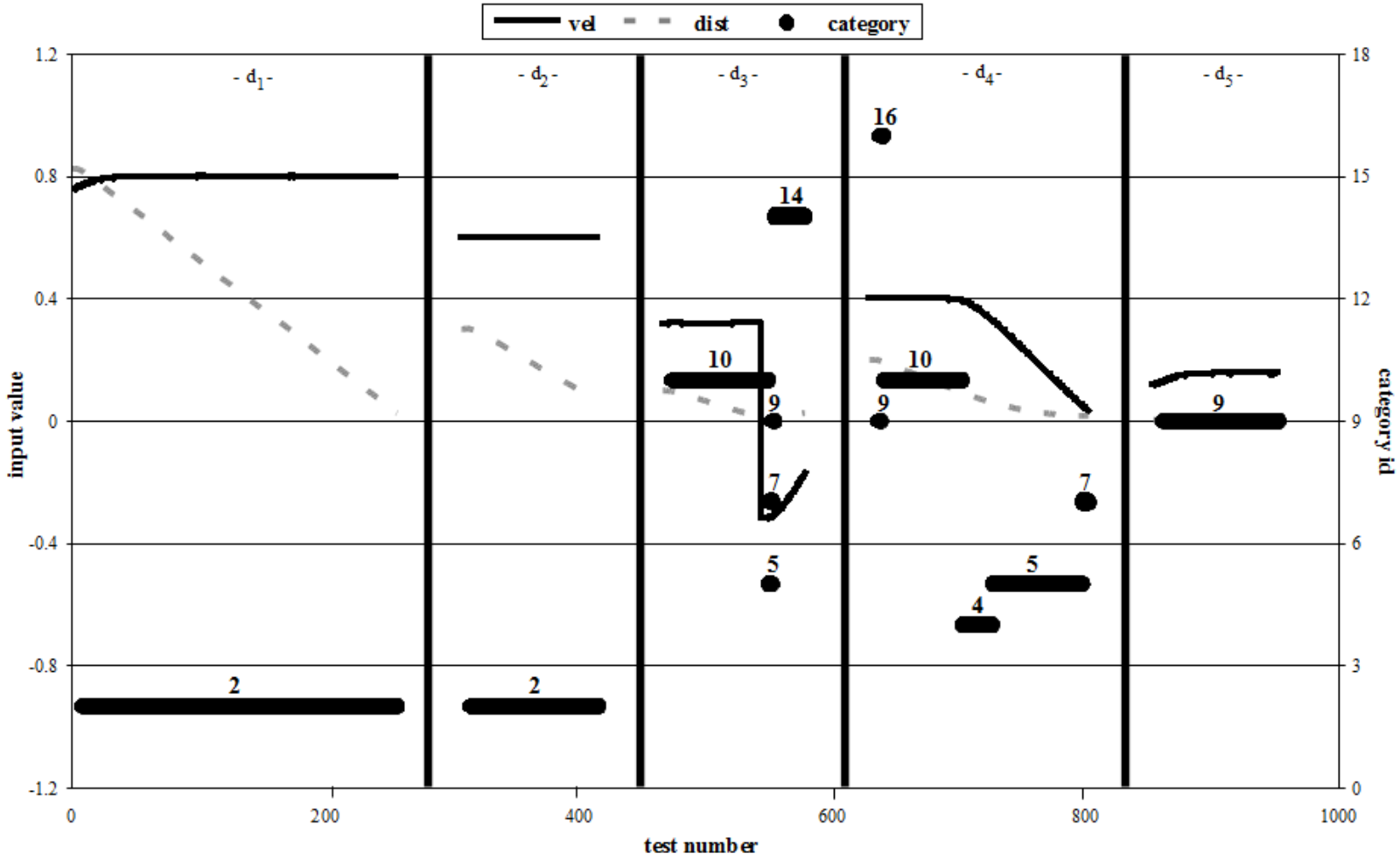

Figure 5.8: Behavior categorization using CobART - 1 [63]

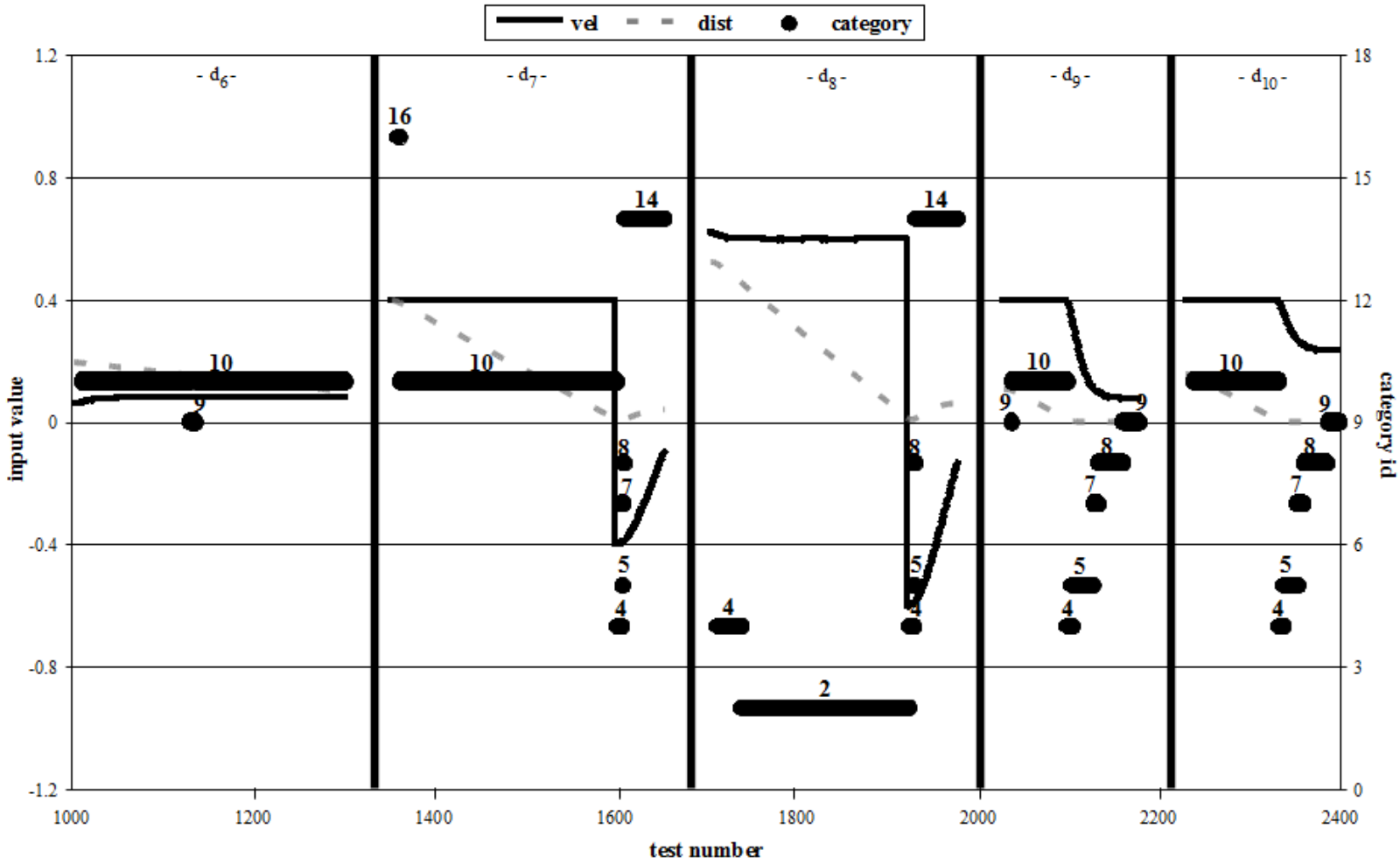

Figure 5.9: Behavior categorization using CobART - 2 [63]

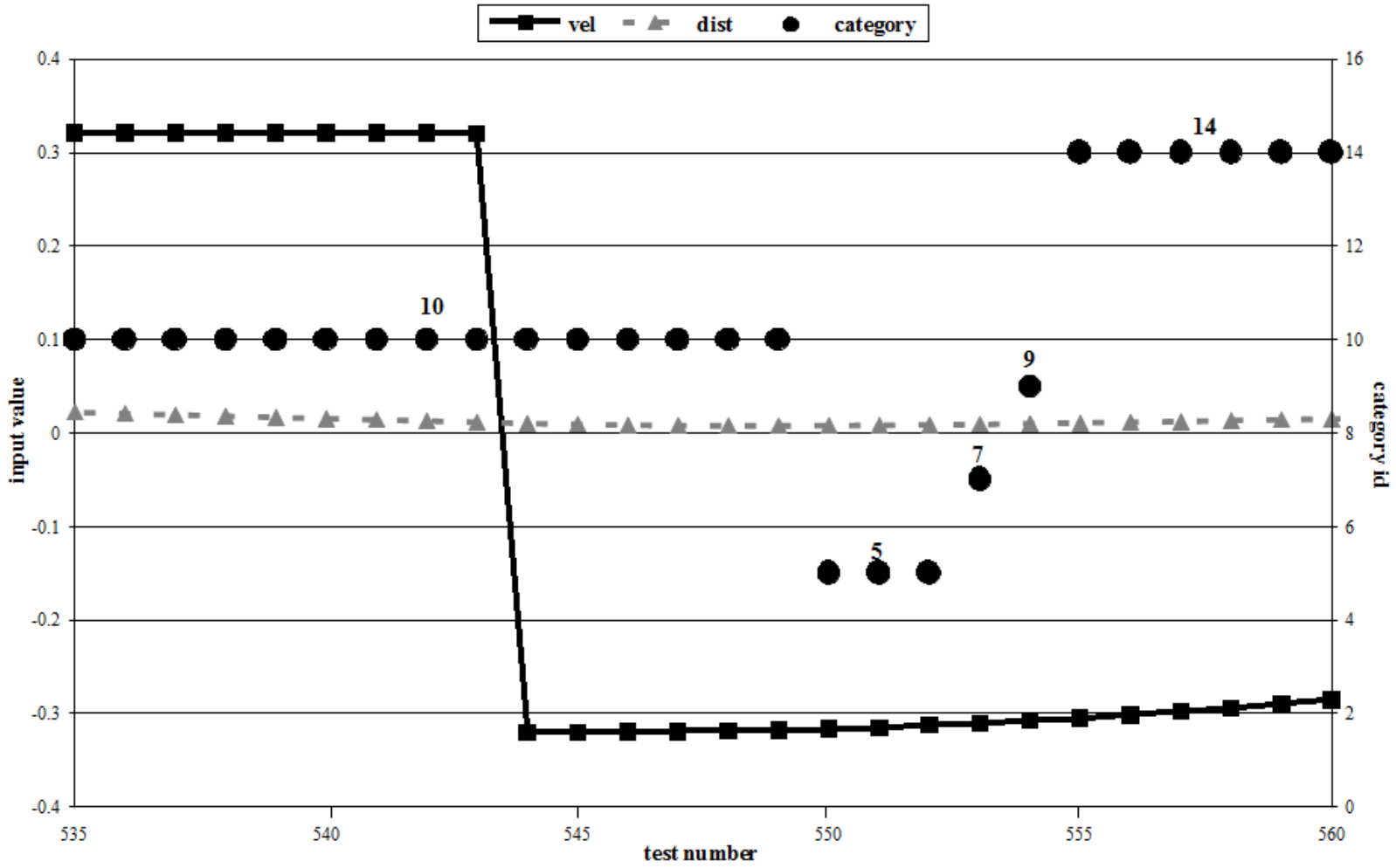

Figure 5.10: Detailed view of CobART behavior categorization [63]

Table 5.3: CobART categories generated for corresponding behaviors [63]

data segments	approaching	pushing	striking
d_1	2	-	-
d_2	2	-	-
d_3 (before 530)	5,10	-	-
d_3 (after 530)	-	-	14
d_4	4,5,7,10	-	-
d_5	-	9	-
d_6	9,10	-	-
d_7 (before 1580)	4,10	-	-
d_7 (after 1580)	-	-	14
d_8 (before 1950)	2,4	-	-
d_8 (after 1950)	-	-	14
d_9 (before 2120)	10	-	-
d_9 (after 2120)	-	4,5,7,8,9	-
d_{10} (before 2320)	10	-	-
d_{10} (after 2320)	-	4,5,7,8,9	-

- Pushing behaviors at data segments d_5, d_9, and d_{10} are classified with categories 8 and 9.

- Going back after striking behaviors at d_3, d_7, and d_8 are classified with category 14.

- Approaching behaviors at d_1, d_2, and d_8 are classified with category 2, while approaching behaviors having slower velocity and shorter distance at d_3, d_4, d_6,d_7, d_9, and d_{10} are classified with category 10.

- Besides, categories 4, 5, and 7 are mapped to the behaviors having decreasing velocities when changing from approaching behavior to pushing behavior.

In the table, the categories being mapped by consecutive data are listed while the ones mapped by an instant data only are omitted (e.g. categories 9 and 16 at d_4). On the other hand, category numbers 0, 1, 3, 6, etc. are missing in the figure. This is because, the categories generated in the learning phase can be used in the testing phase only after they reached the desired saturation. The categories are assumed to be saturated when they are mapped to some behavior and updated for a predefined number of times.

According to the test results, only 80 instances of the 1950 test cases are mapped by the wrong categories. This result shows 95.9% correctness rate. When inappropriate categorizations are analyzed, it is became clear that, significant number of these mistakes are generated during the pass from approaching to pushing or striking behaviors.

The experiments on CobART behavior categorization show that CobART discriminates different behaviors but, in some cases, it categorizes similar behaviors differently [63]. For example, it treats fast approaching and slow approaching behaviors as different categories. Decreasing the vigilance parameter may help generating fewer categories for similar behaviors, but this may result in labeling different behaviors by the same categories.

5.5 Behavior Categorization Using Hierarchically Integrated CobART Networks

Generation of multiple categories for the similar behaviors does not always constitute a problem in the model [63]. Because it is sometimes a need to learn different forms of the behaviors. Nevertheless, the relation between these categories shall be revealed for correct behavior recognition and learning tasks. Hence, we need a method that can differentiate between fast and slow approaching behaviors of the robot, and can understand that both of them are types of the approaching behavior. On the other hand, robot can be performing more than one behavior at the same time. When it is moving, it can be approaching to an object, and it can be moving away from another. In order to elicit relations between these categories and behaviors, a second layer CobART categorization is added and hierarchically integrated CobART networks is constructed as in Figure 5.11.

Hierarchically integrated CobART networks model aims to recognize all the behaviors of the robot by investigating the status of the robot and observing its effects over the objects in the environment [63]. Figure 5.11 depicts the integration of CobART networks that can recognize robot's self behavior (going forward, slowing down, going backward), object's behavior (moving or stopping) and robot's behavior with re-

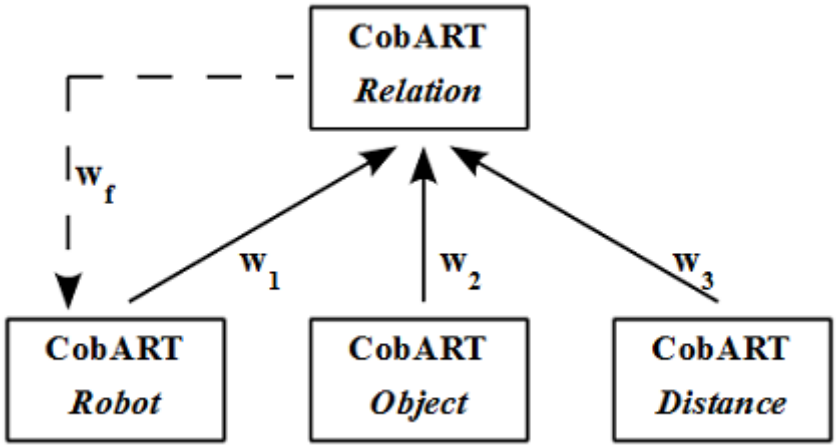

Figure 5.11: Hierarchically integrated CobART networks [63]

spect to the object (approaching, pushing, or striking). This figure can be extended to reveal other behaviors of the robot by integrating the *CobART Robot* network with other CobART networks which categorize motion data of the other objects, motion data of other robots, or moving parts of the robot (e.g. lever) itself. Hence, hierarchical model has an expandable architecture, and it contains reusable parts, which saves training time. Once the training of any first layer network is completed, it can be connected to other hierarchical models for categorization of other behaviors.

In Figure 5.11, *CobART Robot* network receives robot's motion data as an input and generates robot's self behavior categories as an output [63]. Similarly, *CobART Object* network receives object's motion data as an input and generates object's self behavior categories as an output. *CobART Distance* network receives the distance between the robot and the object as an input and categorizes it. *CobART Relation* network at the second layer, categorizes robot's behavior with respect to its effect on an object. To this end, category data from the first layer are fed into the *CobART Relation* network as an input. Hence, categorized robot motion, object motion and distance data are combined and categorized for the second time. Since these data have already been filtered at the first layer CobART networks, filtering at the second layer is bypassed.

First and second layer networks' associations have weight parameters which are labeled as w_1, w_2, and w_3 as shown in the figure [63]. These parameters determine

the weighted averages of the first layer CobART networks. These are used for the matching score calculation at the second layer. Therefore, equation (5.4) is replaced by equation (5.7) for the second layer of hierarchical model. In the equation, M represents the number of CobART networks at the first layer while w_m represents the weight of the m^{th} CobART network, and I_m represents the input vector length for the corresponding CobART network. Besides, p_i and z_{ki} in (5.4) are changed to p_{mi} and z_{mki} to represent processed input data and k^{th} category data for the m^{th} CobART network respectively.

$$r_k = \sum_{m=1}^{M} \left(\frac{w_m}{I_m} \times \sum_{i=1}^{I_m} (DCM(p_{mi}, z_{mki}) \times b + EDM(p_{mi}, z_{mki}) \times (1 - b)) \right) \tag{5.7}$$

In order to reveal the closeness of the generated categories, two types of information are used [63]. First, category data are matched with each other using equation (5.4) for the first layer, and equation (5.7) for the second layer CobART networks at the end of the category learning phase. In order to prevent confusion, these matching scores from equations (5.4) and (5.7) are denoted by r_c and named as correlation score. If the r_c is lower than zero, it is deduced that they are representing totally different behaviors. When the r_c is higher than zero then the categories are similar. The higher the score is the similar the compared categories are.

Second, there is a clipped association between the *CobART Relation* and the *CobART Robot* networks [63]. This association carries matching information to the first layer, and it has a weight parameter w_f (feedback weight). This information is used for detecting the correlation between the first layer CobART categories. This clipped association can be connected to any of the first layer CobART networks in order to find out the category correlations of that network. Our model carries matching information to the *CobART Robot* network. The *CobART Robot* network categories matching to the same categories of the *CobART Relation* network are deduced to have some correlation between them. In order to calculate this correlation, the percentages of mapping to the second layer categories are calculated for each *CobART Robot* network category at the end of the learning phase. When two categories of the first layer CobART network activated the same category on the second layer CobART

network, correlation between these categories is increased. The degree of such correlation is named as matching score and denoted by r_m. When they activated different second layer categories, correlation between the first layer categories is increased or decreased depending on the correlation score between the activated second layer categories (r_c>0 or r_c<0).

The feedback weight w_f and the matching score r_m are used to revise the first layer category correlation score r_c according to equation (5.8) [63]. In the equation, f_1 and f_2 correspond to the first layer categories whose correlation score is being calculated. The feedback weight is in the range of [0:1]; it starts from an initial value and decreases gradually to the zero in order to protect learned networks from endless changes as in equation (4.4) which represents CobART learning rate update rule. This parameter is kept with the first layer CobART network that receives the feedback from the second layer, and is decreased according to the weight update rule each time that CobART network is used for the learning of any hierarchical model.

$$r_c(f_1, f_2) = r_c(f_1, f_2) + r_m(f_1, f_2) \times w_f \tag{5.8}$$

Equation (5.9) is used for matching score calculations of the first layer categories f_1 and f_2 [63]. In the equation, s_1 and s_2 correspond to any two categories from the second layer, and K represents the number of the categories in that layer. A function labeled as *mp* corresponds to a mapping percentage from a first layer category to a second layer category. Mapping percentage between f_1 and s_1 is simply calculated by dividing the number of concurrent f_1 and s_1 activations by the total number of f_1 activations. Calculation of the r_m can be better explained with an example. Assume *CobART Robot* network category f_1 is activated with the second layer category s_1 in the 100% of its activations. Besides, *CobART Robot* network category f_2 is activated with s_1 in the 60% and with s_2 in the 40% of its activations. It is also assumed that the correlation score r_c between s_1 and s_2 is 0.25. Since both f_1 and f_2 are activated with s_1, r_m is set to the 0.6 (60% s_1 activations in common). Then, this value is increased by 0.1 because of s_1 and s_2 activations (different category activation percentage 40% is multiplied by the correlation score 0.25). As a result, r_m is calculated as 0.7. For

this example, when w_f is initialized to 0.2, r_c is increased by 0.14 (0.7×0.2). At the end, if r_c exceeds the selected vigilance parameter ρ, the categories f_1 and f_2 can be assumed to represent the same behavior.

$$r_m(f_1, f_2) = \sum_{s_1=1}^{K} \sum_{s_2=1}^{K} (mp(f_1, s_1) \times mp(f_2, s_2) \times r_c(s_1, s_2)) \tag{5.9}$$

5.5.1 Hierarchical Model Evaluation

Performance of the hierarchical model for behavior categorization is initially tested by feeding the *CobART Robot* network (see Figure 5.11) with the robot's velocity and position, the *CobART Object* network with the target object's velocity and position, and the *CobART Distance* network with the distance between the robot and the object [63]. Although reasonable categorization is observed (see Appendix A), the categorization is also tested by extracting the robot's position and the object's position from the input data. Figure 5.12 and Figure 5.13 show that reasonable categorization is also achieved when the position data is not used as an input. This categorization process is given in equation (5.10). The *CobART Robot* network of the hierarchical model receives the robot's velocity data as an input and generates the robot's self behavior categories as an output. The equation represents the *CobART Robot* network categorization process at time t with c_t^r symbol. In the equation, N and vel^r correspond to the consecutive input number and the robot velocity data, respectively. Similarly, the *CobART Object* network receives the object's velocity data as an input and generates the object's self behavior categories as an output. The equation represents the *CobART Object* network categorization process with c_t^o symbol. In the equation, vel^o corresponds to the object velocity data. Last network of the first layer, the *CobART Distance*, receives the distance between the robot and the object as an input and categorizes it. This categorization is denoted by c_t^d, for time t in the equation, and the distance data is represented as *dist*. The *CobART Relation* network at the second layer categorizes the robot's behavior with respect to its effect on the target object. Category data from the first layer are fed into the *CobART Relation* network as an input. Hence, categorized robot velocity, object velocity and distance data are combined

and categorized for the second time. The *CobART Relation* network categorization is denoted by c_t in the equation.

$$\begin{aligned} c_t^r &= CobART(vel_t^r, .., vel_{t-N+1}^r) \\ c_t^o &= CobART(vel_t^o, .., vel_{t-N+1}^o) \\ c_t^d &= CobART(dist_t, .., dist_{t-N+1}) \\ c_t &= CobART(c_t^r, c_t^o, c_t^d) \end{aligned} \tag{5.10}$$

In Figure 5.12 and Figure 5.13, just the robot velocity and the distance to the object are given along with the activated categories in order not to complicate the graph [63]. This categorization is obtained when α is -1.5, d is 0.2, a is 0.8, b is 0.7, and N is 10. The vigilance parameter ρ is set to 0.8 for the first layer and 0.6 for the second layer networks. A high value of ρ is used for the first layer networks in order to reveal the different forms of primitive behaviors. However, a lower ρ value is used for the second layer network to find out the close correlation between these behaviors, and to generate fewer categories that can discriminate different behaviors adequately. On the other hand, weight parameters w_1, w_2, and w_3 are set to 0.3, 0.3, and 0.4 respectively, in order to make the effect of the *CobART Distance* network higher than effects of the other networks. The feedback weight parameter w_f which will be used to find close correlation between the categories is initialized to 0.2.

Table 5.4 summarizes categorization results of the hierarchical model, and presents the mapping between the behaviors and generated categories for each data segment [63]. Figure 5.12, Figure 5.13 and Table 5.4 show that;

- Approaching behaviors are classified with the categories 0 and 1. Category 0 corresponds to an approaching behavior at higher velocity with longer distance when compared to category 1.

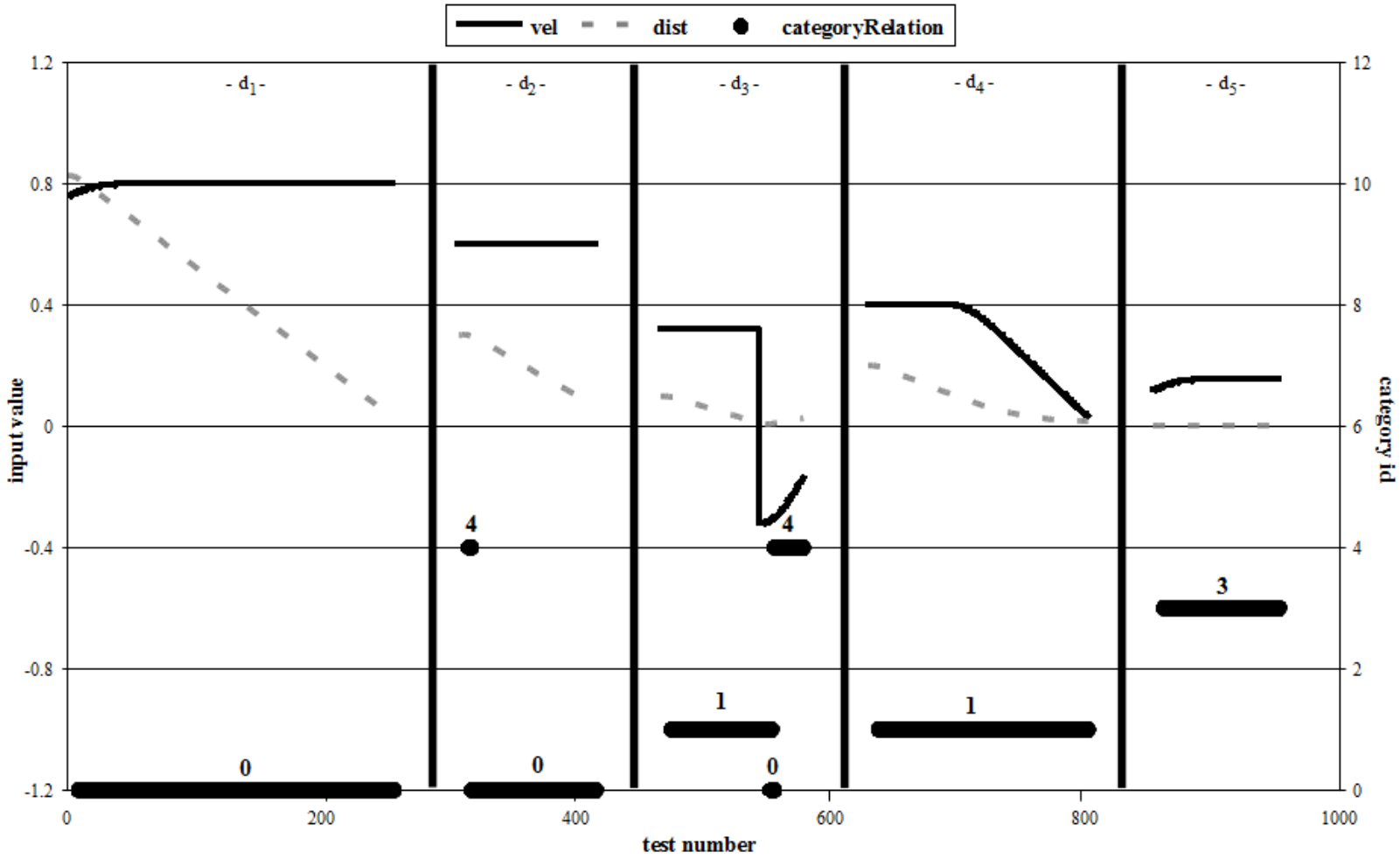

Figure 5.12: Behavior categorization by hierarchical model - 1 [63]

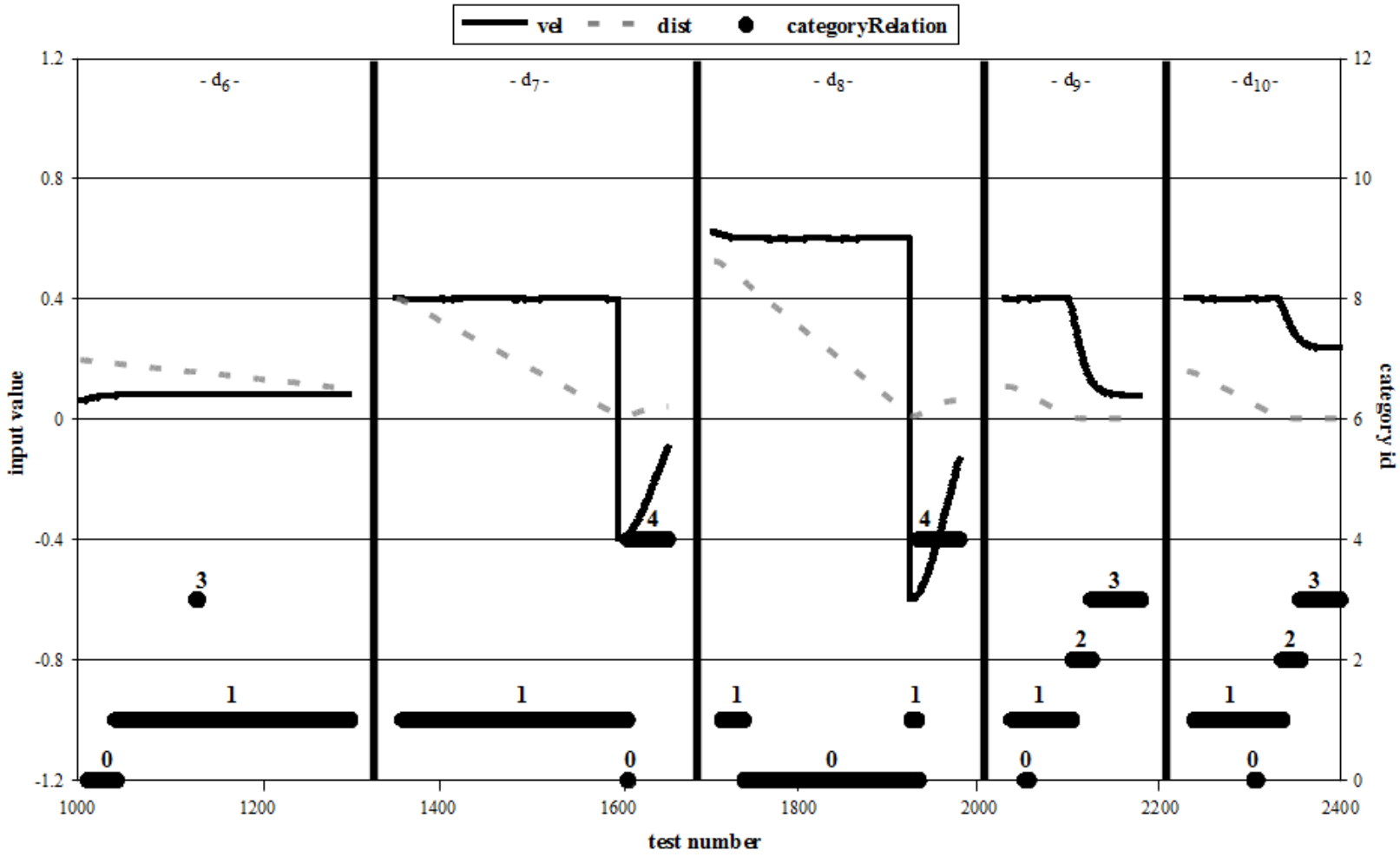

Figure 5.13: Behavior categorization by hierarchical model - 2 [63]

Table 5.4: *CobART Relation* categories generated for corresponding behaviors [63]

data segments	approaching	pushing	striking
d_1	0	-	-
d_2	0,4	-	-
d_3 (before 530)	0,1	-	-
d_3 (after 530)	-	-	4
d_4	1	-	-
d_5	-	3	-
d_6	0,1,3	-	-
d_7 (before 1580)	1	-	-
d_7 (after 1580)	-	-	4
d_8 (before 1950)	0,1	-	-
d_8 (after 1950)	-	-	4
d_9 (before 2120)	0,1	-	-
d_9 (after 2120)	-	2,3	-
d_{10} (before 2320)	0,1	-	-
d_{10} (after 2320)	-	2,3	-

- Pushing behaviors at data segments d_5, d_9, and d_{10} are classified with categories 2 and 3. Category 3 is mapped to a pushing behavior having more stable velocity when compared to category 2.

- Going back after striking behaviors at d_3, d_7, and d_8 are classified with category 4.

- On the other hand, categories 3 and 4 are also mapped to the approaching behaviors at d_2 and d_6 temporarily, for a few input samples.

According to the test results, only 53 instances of the 1950 test cases are mapped by the wrong categories. This results in 97.3% correctness rate. The significant number of these mistakes are generated during the pass from approaching to pushing or striking behaviors. Although a small number of inappropriate mapping shows the success of the model, they can be eliminated by increasing the vigilance parameter of the first or the second layer. However, such a change increases category number and makes the analysis complicated.

These results show that, hierarchically integrated CobART network model can classify behavior data into more generic and correct categories, and hence outperforms

Table 5.5: *CobART Relation* network correlation score r_c [63]

categories	s_0	s_1	s_2	s_3	s_4
s_0	-	0.54	0.25	-0.06	0.32
s_1	0.54	-	0.54	0.3	0.19
s_2	0.25	0.54	-	0.52	-0.14
s_3	-0.06	0.3	0.52	-	0.2
s_4	0.32	0.19	-0.14	0.2	-

Table 5.6: *CobART Robot* categories generated for corresponding behaviors [63]

data segments	going forward	going backward
d_1	2,3	-
d_2	2	-
d_3 (before 530)	8,9	-
d_3 (after 530)	-	16
d_4	6,7,9	-
d_5	3,9,16	-
d_6	3,9	-
d_7 (before 1580)	9	-
d_7 (after 1580)	-	16
d_8 (before 1950)	2,3,8	-
d_8 (after 1950)	-	16
d_9 (before 2120)	9	-
d_9 (after 2120)	5,6,7,8,9	-
d_{10} (before 2320)	9	-
d_{10} (after 2320)	5,6,7,8,9	-

single CobART network in behavior categorization process. The correlation between these *CobART Relation* network categories are calculated at the end of the learning phase according to equation (5.7), and the results are represented in Table 5.5. In the table, the second layer categories are labeled as s_0, s_1, s_2, s_3, and s_4. The correlation scores show that there is some correlation between s_0 and s_1, s_1 and s_2, and s_2 and s_3.

Figure 5.14 and Figure 5.15 represent the robot's self behavior categorization based on its velocity data, and Table 5.6 shows the mapping between the behaviors and generated categories for each data segment [63]. It results in having 8 types of categories. This categorization process is given in equation (5.10) with c_t^r. The categorization results show that;

- Going forward behavior is classified with categories 2, 3, 5, 6, 7, 8, and 9.

Table 5.7: *CobART Robot* network correlation score r_c [63]

categories	f_2	f_3	f_5	f_6	f_7	f_8	f_9	f_{16}
f_2	-	0.62	0.16	0.11	0.14	0.4	0.79	-0.02
f_3	0.62	-	-0.04	-0.1	-0.06	0.23	0.62	0.4
f_5	0.16	-0.04	-	0.74	0.65	0.55	0.22	-0.42
f_6	0.11	-0.1	0.74	-	0.78	0.62	0.28	-0.45
f_7	0.14	-0.06	0.65	0.78	-	0.75	0.32	-0.28
f_8	0.4	0.23	0.55	0.62	0.75	-	0.61	-0.15
f_9	0.79	0.62	0.22	0.28	0.32	0.61	-	0.19
f_{16}	-0.02	0.4	-0.42	-0.45	-0.28	-0.15	0.19	-

- Going backward is classified with category 16.
- Category 2 is mapped by the going forward behavior having stable and high velocity, while category 9 mapped by the stable and lower velocity behavior.
- Categories 5 through 8 are mapped by the slowing down going forward behaviors while category 3 is generally mapped by the speeding up going forward behavior.

According to the test results on *CobART Robot* network, only 66 instances of the 1950 test cases are mapped by the wrong categories. This results in 96.6% correctness rate.

Correlation between *CobART Robot* network categories are calculated at the end of the learning phase according to equation (5.4), and represented in Table 5.7 with the labels $f_2, f_3, f_5, f_6, f_7, f_8, f_9$, and f_{16} corresponding to the categories in Figure 5.14 and Figure 5.15 [63]. The correlation scores in the table show that there exists close correlation between f_2 and f_9, f_5 and f_6, f_6 and f_7, and f_7 and f_8.

As explained before, the clipped association from the *CobART Relation* network to the *CobART Robot* network carries the matching information, and this information is also used to determine the degree of similarity between the first layer categories [63]. The percentages of the second layer category mappings are stored for each first layer category to calculate such a correlation. Table 5.8 shows these values for *CobART Robot* network categories. The table shows that, 100% of the f_2 activations were

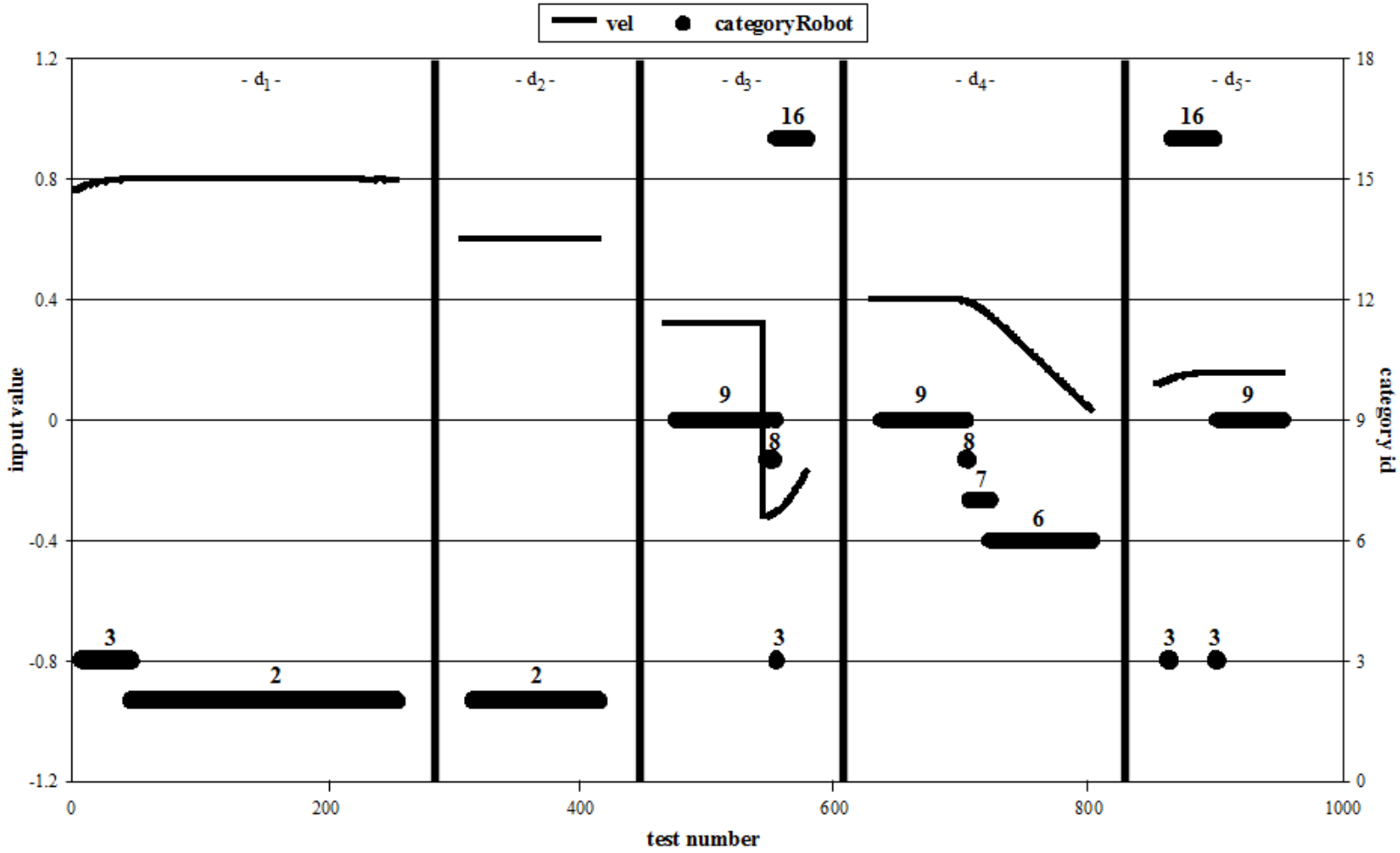

Figure 5.14: Robot self behavior categorization by CobART Robot network - 1 [63]

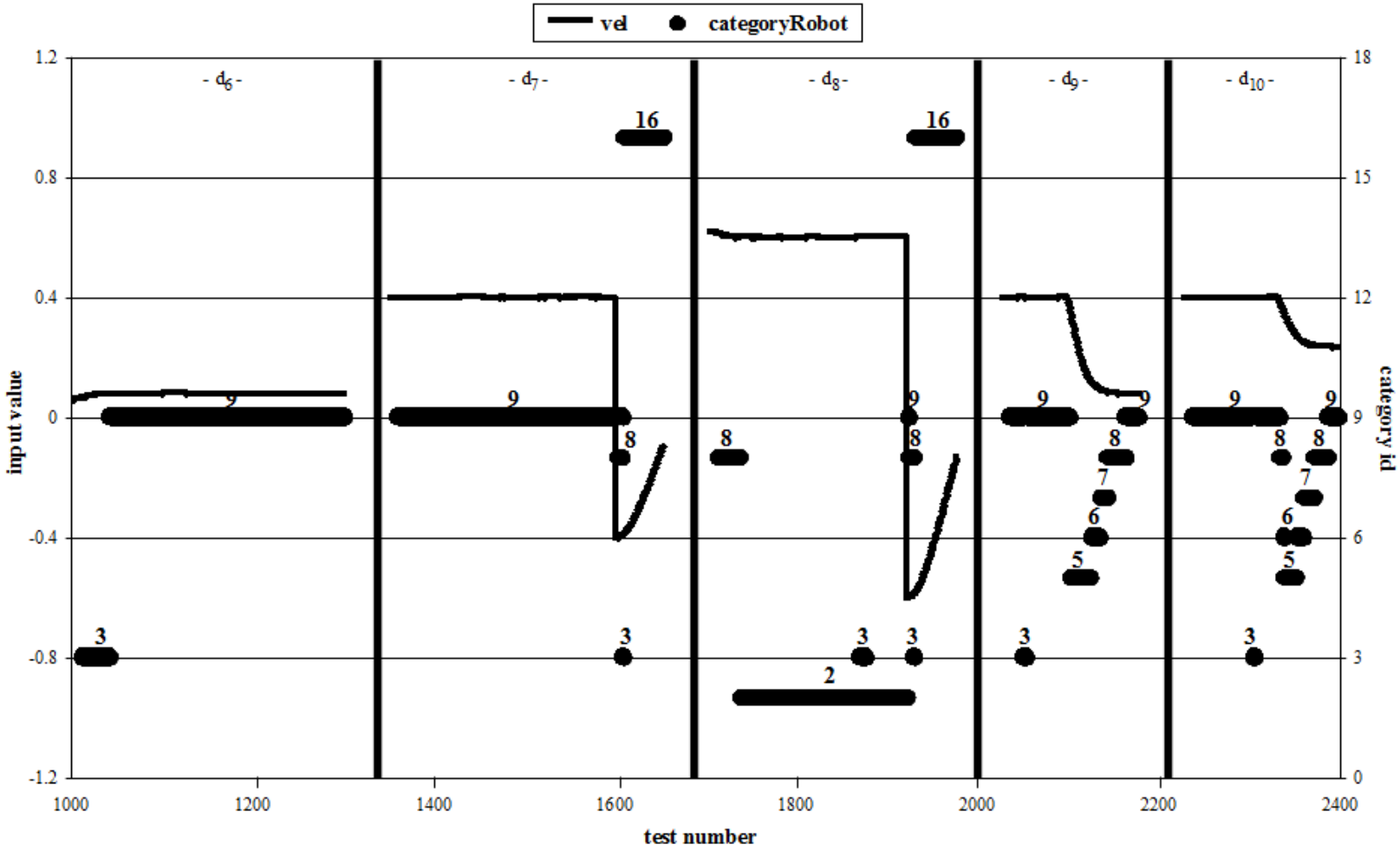

Figure 5.15: Robot self behavior categorization by CobART Robot network - 2 [63]

Table 5.8: Second layer category mapping percentages for *CobART Robot* categories [63]

categories	s_0	s_1	s_2	s_3	s_4
f_2	1.0	-	-	-	-
f_3	0.99	-	-	-	0.01
f_5	-	-	0.92	0.08	-
f_6	-	0.67	0.06	0.27	-
f_7	-	0.44	0.01	0.55	-
f_8	-	0.32	0.03	0.65	-
f_9	-	0.56	0.01	0.43	-
f_{16}	0.02	-	-	-	0.98

resulted in s_0. On the other hand, 67% of the f_6 activations were resulted in s_1, 6% were in s_2, and 27% were in s_3.

Table 5.9 displays category matching scores (r_m) between the *CobART Robot* network categories which is calculated according to equation (5.9) by using the values from Table 5.5 and Table 5.8 [63]. For instance, categories f_2 and f_3 have common 99% percentage of s_0 mapping, so r_m between f_2 and f_3 is 0.99 as listed in Table 5.9. Calculation of the r_m between f_2 and f_6 is as follows: Table 5.8 shows that these two categories are mapped to 4 types of second layer categories, having no common mapping. Category f_2 is mapped to s_0 at all its activations, while category f_6 is mapped to the categories s_1, s_2, and s_3. Therefore s_0 and s_1, s_0 and s_2, and s_0 and s_3 activation information are used to calculate r_m. Starting from s_0 and s_1 activations, r_m is initialized to 0.36 (the result of 1.0×0.67×0.54). 1.0 and 0.67 are the mapping percentages from f_2 and f_6 to s_0 and s_1 according to Table 5.8, and 0.54 is the correlation score between s_0 and s_1 according to Table 5.5. Then, matching score of 0.02 (the result of 1.0×0.06×0.25) is added and decreased by 0.02 (the result of 1.0×0.27×(-0.06)) for s_0 and s_2, and s_0 and s_3 activations. Consequently, r_m between f_2 and f_6 equals to 0.36 as listed in Table 5.9. The multiplication of this r_m value with the feedback weight, which is set to 0.2, causes correlation between these categories to be increased by 0.07 (the result of 0.36×0.2) according to equation (5.8). Table 5.7 and Table 5.10 show that r_c between f_2 and f_6 is increased from 0.11 to 0.18.

Table 5.9: *CobART Robot* network matching score r_m [63]

categories	f_2	f_3	f_5	f_6	f_7	f_8	f_9	f_{16}
f_2	-	0.99	0.23	0.36	0.2	0.14	0.28	0.34
f_3	0.99	-	0.22	0.36	0.2	0.14	0.28	0.34
f_5	0.23	0.22	-	0.56	0.55	0.56	0.54	-0.11
f_6	0.36	0.36	0.56	-	0.62	0.59	0.66	0.18
f_7	0.2	0.2	0.55	0.62	-	0.66	0.64	0.19
f_8	0.14	0.14	0.56	0.59	0.66	-	0.63	0.19
f_9	0.28	0.28	0.54	0.66	0.64	0.63	-	0.19
f_{16}	0.34	0.34	-0.11	0.18	0.19	0.19	0.19	-

Table 5.10: *CobART Robot* network adjusted correlation score r_c [63]

categories	f_2	f_3	f_5	f_6	f_7	f_8	f_9	f_{16}
f_2	-	0.82	0.21	0.18	0.18	0.43	0.84	0.05
f_3	0.82	-	0	-0.02	-0.02	0.26	0.68	0.47
f_5	0.21	0	-	0.85	0.76	0.66	0.33	-0.44
f_6	0.18	-0.02	0.85	-	0.91	0.74	0.41	-0.41
f_7	0.18	-0.02	0.76	0.91	-	0.88	0.45	-0.25
f_8	0.43	0.26	0.66	0.74	0.88	-	0.74	-0.11
f_9	0.84	0.68	0.33	0.41	0.45	0.74	-	0.23
f_{16}	0.05	0.47	-0.44	-0.41	-0.25	-0.11	0.23	-

Table 5.10 contains the adjusted values of the correlation scores [63]. According to the table, correlation between f_2 and f_3, f_2 and f_9, f_5 and f_6, f_6 and f_7, and f_7 and f_8 are higher than the vigilance parameter and they can be grouped under the same behavior. Besides, adjusted correlation between f_5 and f_7, f_6 and f_8, and f_8 and f_9 are close to the vigilance parameter, and it shows high similarity between them.

Eventually, to illustrate the effectiveness of the proposed model clearly, number of the categories are hold at the limit by using a relatively small vigilance parameter in the experiments [63]. In reality, robot should learn more variations of the similar behaviors. This can be done by using a relatively high vigilance parameter, and by performing much longer training. In fact, learning is a continuing process; when the

CobART Robot network is integrated with other types of CobART networks in order to reveal some other behaviors, the *CobART Robot* network should also be updated by adjusting Table 5.10 based on the matching information from that integration.

CHAPTER 6

DISCUSSION & CONCLUSION

Behavior categorization models are presented to elicit alternative forms of primitive behaviors and to find relations between them. Behavior categorization by ART 2 network was implemented and tested first, for its unsupervised and self-organizing nature. A wide range of tests on the ART 2 parameters were not resulted in reasonable categorization; and priori SOM categorization was combined with the ART 2. Because this new architecture did not enhance the behavior categorization results, a new type of ART 2 network is developed.

CobART is a new type of unsupervised, self-organizing stable category generation network that uses correlation analysis methods and different weight update rule [62]. Its performance, which is based on 50 types of analog input pattern categorization, is compared with that of ART 2 experimentally. Experiments showed that, CobART produces more reasonable categorization than ART 2.

Another advantage of CobART is its easy-to-change structure [62]. For the categorization of specific types of analog input patterns, related correlation analysis methods can be added to the network easily whenever needed. This change ensures that, the resultant categories are generated according to new correlational matching.

After that, CobART is applied to behavior categorization task, and tested experimentally. The model is tested on behaviors such as *pushing*, *approaching*, and *striking*. The test results show that CobART can discriminate different behaviors adequately [63]. However, it generates more than one category for similar behaviors depending on the shape of the behaviors. Generation of multiple categories for the similar be-

haviors does not always constitute a problem in the model. Because it is sometimes a need to learn different forms of the behaviors. Nevertheless, the relation between these categories shall be revealed for correct behavior recognition and learning tasks. Hence, it is a need to differentiate between fast and slow motion of the robot, and to understand that both of them are types of the same behavior.

CobART networks are integrated hierarchically in order to improve the categorization performance, and to elicit various behaviors performed by the robot at the same time [63]. In the first layer, CobART networks categorize the motion data of the robot or objects. In the second layer, CobART network is fed by the category outputs of the first layer networks and categorizes them. As a result, robot's behavior with respect to an object is generated by the second layer CobART network. It can be observed from the experiments that hierarchically integrated CobART networks model produces more reasonable categorization results than others, and reaches up to 97.3% correctness rate.

In order to reveal the closeness of the generated categories, correlation between the first layer categories are calculated by using correlation analysis methods used at the categorization process, and by using the feedback from the second layer network to the first layer networks. This feedback carries matching information. The first layer categories matching to the same categories on the second layer network are deduced to have some correlation between them [63]. In this case, correlation score between the corresponding first layer categories is increased. Similarly, when they activated different second layer categories, correlation between the corresponding first layer categories is increased or decreased depending on the correlation score between the activated second layer categories. When the resultant correlation score exceeds the vigilance parameter, the corresponding first layer categories can be tagged as the alternative forms of the same behaviors.

The hierarchically integrated CobART networks model presents a way to reveal all behaviors performed by the robot at the same time [63]. To this end, the model can be expanded by adding new CobART networks to the first or the second layers of the hierarchical model. The new CobART network in the first layer may categorize motion data of a specific component of the robot, or motion data of the another object,

or motion data of another robot, while the new CobART network in the second layer may categorize robot's behavior with respect to its effect on another object or another robot moving in the environment. Thus, hierarchical model has an expandable architecture, and it contains reusable parts [63]. Once the training of any first layer network is completed, it can be connected to other hierarchical models for categorization of other behaviors.

The studies on behavior categorization task is started with the idea of developing a generic, unsupervised and self-organizing behavior model. This idea is inspired by the insufficient studies in the related areas. Besides, they generally have assumptions over the environment, over the learning method, or over the task. The most exciting paper for me was Tani's study [56]. Tani presented a novel hierarchical neural network for behavior learning. Model includes two layer Recurrent Neural Network running on different time scales. The lower level RNN learns primitive behaviors by using sensory and motor data. On the other hand, the top level RNN learns the sequences between the primitive behaviors and time spent at each of them. Test results on robot arm show that, the robot can regenerate learned behaviors without significant discrepancies when order of the primitive behaviors selected according to design rules.

That study is successful at generation of the learned behaviors. However, the model has some weaknesses when compared to our study. First, all primitive behaviors share the same RNN. Addition of new primitive behaviors causes updates over the shared weight vectors. These updates may affect correctness of already learned primitive behaviors. Second, the model is tested with designed sequence of the primitive behaviors, it is not tested with the alternative forms of the primitive behaviors. When slower or faster form of the primitive behaviors are presented to network, more discrepancies can be seen during the generation of these behaviors. Third, when another component of the robot is added to the model in order to learn combined behaviors, the RNN must need to be redesigned which will cause already learned information to be lost.

At long last, the hierarchical model can be used for behavior recognition of robots and objects, and it can help construction of a base model for behavior learning and

generation [63]. For instance, once a robot learns primitive behaviors of navigation and grasping behaviors, it can learn carrying an object behavior more easily. On the other hand, robots can use such a system to recognize friends' or opponents' behaviors in group actions or in games to behave accordingly. When a robot is asked to perform a specific task, it can select appropriate primitive behaviors depending on the environment and its current behavior status. Moreover, once a robot learns effects of the behaviors, it can guess effects of the similar behaviors using such a model. For example, an observer that watches machines in a factory can detect potentially abnormal conditions once it learns all forms of the machine operations. On the other hand, single CobART network can provide many contributions to the studies in various sciences [62]. For example, it may be used at sound recognition in a way such that it can discriminate simple sound signals effectively. Moreover, CobART can contribute to the studies in biomedical engineering. For example, ECG (electrocardiograph) signals can be categorized into the subcategories that can represent various normal and potentially abnormal heart rhythms.

REFERENCES

[1] E.A. Billing and T. Hellstrom. Behavior recognition for segmentation of demonstrated tasks. pages 228–234. IEEE SMC International Conference on Distributed Human-Machine Systems, 2008.

[2] J. Blynel and D. Floreano. Levels of dynamics and adaptive behavior in evolutionary neural controllers. pages 272–281. ICSAB, Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior on From Animals to Animats, 2002.

[3] E. Borenstein and E. Ruppin. The evolution of imitation and mirror neurons in adaptive agents. *Cognitive Systems Research*, 6:229–242, 2005.

[4] G.A. Carpenter and S. Grossberg. Art 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, 1987.

[5] G.A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[6] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, 1992.

[7] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, 4:565–588, 1991.

[8] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 4:493–504, 1991.

[9] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[10] M. Caudill and C. Butler. *Naturally Intelligent Systems*. The MIT Press, Cambridge, MA, 1990.

[11] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[12] P.M. Embree and B. Kimble. *C language algorithms for digital signal processing*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[13] M. Fox, M. Ghallab, G. Infantes, and D. Long. Robot introspection through learned hidden markov models. *Artificial Intelligence*, 170:59–113, 2006.

[14] T. Frank, K. F. Kraiss, and T. Kuhlen. Comparative analysis of fuzzy art and art-2a network clustering performance. *IEEE Transactions on Neural Networks*, 9(3):544–559, 1998.

[15] W.K. Fung and Y.H. Liu. Adaptive categorization of art networks in robot behavior learning using game-theoretic formulation. *Neural Networks*, 16:1403–1420, 2003.

[16] SNNS Group. Stuttgart neural network simulator (snns) user manual, version 4.2. *University of Tubingen, Tubingen*, 1998.

[17] L. Gu and J. Su. Humanoid robot behavior learning based on art neural network and cross-modality learning. pages 447–450. ICNC, 2006.

[18] M.O. Halloran, B. McGinley, R.C. Conceicao, F. Morgan, E. Jones, and M. Glavin. Spiking neural networks for breast cancer classification in a dielectrically heterogeneous breast. *Progress In Electromagnetics Research*, 113:413–428, 2011.

[19] K. Han and M. Veloso. Automated robot behavior recognition applied to robotic soccer. Proceedings of the IJCAI-99 Workshop on Team Behaviors and Plan Recognition, 1999.

[20] S. Haykin. *Neural Networks and Learning Machines*. Pearson Prentice Hall, Upper Saddle River, NJ, 2009.

[21] J.A. Hertz, A. S. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*. Addison Wesley, Reading, MA, 1994.

[22] T. Inamura, Y. Nakamura, H. Ezaki, and I. Toshima. Imitation and primitive symbol acquisition of humanoids by the integrated mimesis loop. pages 4208–4213. Proceedings of the 2001 IEEE International Conference on Robotics & Automation, 2001.

[23] G. Infantes, M. Ghallab, and F. Ingrand. Learning the behavior model of a robot. *Autonomous Robots*, 30(2):157–177, 2011.

[24] M. Ito and J. Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12:93–115, 2004.

[25] K. Itoh, H. Miwa, H. Takanobu, and A. Takanishi. Application of neural network to humanoid robots–development of co-associative memory model. *Neural Netwroks*, 18:666–673, 2005.

[26] D.M. Skapura J.A. Freeman. *Neural networks algorithms, applications, and programming techniques*. Addison Wesley, Reading, MA, 1992.

[27] G. James and R.C. James. *Mathematics Dictionary*. D. Van Nostrand Company, New York, 1959.

[28] J.M. Jenkins and S.A. Caswell. Detection algorithms in implantable cardioverter defibrillators. volume 84, pages 428–445. Proceedings of the IEEE, 1996.

[29] K-Team. Khepera user manual, version 5.02. *http://www.k-team.com*, 1999.

[30] T. Kanda, H. Ishiguro, M. Imai, and T. Ono. Development and evaluation of interactive humanoid robots. volume 92, pages 1839–1850. Proceedings of the IEEE, 2004.

[31] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. Understanding human intentions via hidden markov models in autonomous mobile robots. pages 367–374. HRI, Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction, 2008.

[32] N. Koenig and M. Mataric. Demonstration-based behavior and task learning. AAAI Spring Symposium To Boldy Go Where No Human-Robot Team Has Gone Before, Working notes, 2006.

[33] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[34] N. Kubota. Computational intelligence for structured learning of a partner robot based on imitation. *Information Sciences*, 171:403–429, 2005.

[35] The LAMI (Microprocessor Systems Laboratory). The khepera miniature mobile robot. *http://diwww.epfl.ch/lami/robots/K-family/Khepera.html*.

[36] O. Lebeltel, P. Bessiere, J. Diard, and E. Mazer. Bayesian robot programming. *Autonomous Robots*, 16:49–79, 2004.

[37] L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. AAAI, Proceedings of the National Conference on Artificial Intelligence, 2004.

[38] Cyberbotics Ltd. Webots user guide, version 3.2.24. *http://www.cyberbotics.com*, 2003.

[39] G.F. Luger. *Artificial Intelligence Structures and Strategies for Complex Problem Solving*. Addison Wesley, Harlow, England, 2002.

[40] P.A. Lynn and W. Fuerst. *Introductory digital signal processing with computer applications*. John Wiley & Sons, Chichester, England, 1999.

[41] R.G. Lyons. *Understanding digital signal processing*. Addison Wesley, Reading, MA, 1997.

[42] The MathWorks. Matlab 7 getting started guide. *http://www.mathworks.com*, 2008.

[43] C.J. Merz, D.C. Clair, and W.E. Bond. Semi-supervised adaptive resonance theory (smart2). volume 3, pages 851–856. International Joint Conference on Neural Networks, 1992.

[44] T.M. Mitchell. *Machine Learning*. The McGraw Hill, New York, 1997.

[45] B. Morisset and M. Ghallab. Learning how to combine sensory-motor functions into a robust behavior. *Artificial intelligence*, 172:392–412, 2008.

[46] D. Nau, T.C. Au, O. Ilghami, U. Kuter, J.W. Murdock, D. Wu, and F. Yaman. Shop2: An htn planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[47] M.N. Nicolescu and M.J. Mataric. A hierarchical architecture for behavior-based robots. pages 227–233. AAMAS, Proceedings of the first international joint conference on Autonomous agents and multiagent systems, 2002.

[48] T. Oates, M.D. Schmill, and P.R. Cohen. A method for clustering the experiences of a mobile robot that accords with human judgments. pages 846–851. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, 2000.

[49] A.V. Oppenheim, A.S. Willsky, and S.H. Nawab. *Signals & Systems*. Prentice Hall, Saddle River, NJ, 1997.

[50] S. Osentoski, V. Manfredi, and S. Mahadevan. Learning hierarchical models of activity. pages 891–896. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.

[51] R.W. Paine and J. Tani. Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Networks*, 17:1291–1309, 2004.

[52] R. Palm and B. Iliev. Grasp recognition by time-clustering, fuzzy modeling, and hidden markov models (hmm) - a comprative study. pages 599–605. IEEE International Conference on Fuzzy Systems, 2008.

[53] R. Palm, B. Kadmiry, B. Iliev, and D. Driankov. Recognition and teaching of robot skills by fuzzy time-modeling. pages 7–12. IFSA-EUSFLAT 2009, 2009.

[54] J.M. Peula, C. Urdiales, I. Herrero, I.S. Tato, and F. Sandoval. Pure reactive behavior learning using case based reasoning for a vision based 4-legged robot. *Robotics and Autonomous Systems*, 57:688–699, 2009.

[55] H. Shatkay and L.P. Kaelbling. Learning geometrically-constrained hidden markov models for robot navigation: Bridging the topological-geometrical gap. *Journal of Artificial Intelligence Research*, 16:167–207, 2002.

[56] J. Tani. Learning to generate articulated behavior through the bottom-up and the top-down interaction processes. *Neural Networks*, 16:11–23, 2003.

[57] R.D. Throne and J.M. Jenkins L.A. DiCarlo. A comparison of four new time-domain techniques for discriminating monomorphic ventricular tachycardia from sinus rhythm using ventricular waveform morphology. *IEEE Transactions On Biomedical Engineering*, 38:561–570, 1991.

[58] W.P. Vogt. *Quantitative research methods for professionals*. Pearson, Boston, MA, 2007.

[59] S. Vosniadou and A. Ortony. *Similarity and analogical reasoning*. Cambridge University Press, New York, 1989.

[60] S.X. Yang and H. Li. An autonomous mobile robot with fuzzy obstacle avoidance behaviors and a visual landmark recognition system. pages 1579–1584. ICARCV, Seventh International Conference on Control, Automation, Robotics And Vision, 2002.

[61] M. Yavas and F.N. Alpaslan. Behavior categorization using correlation based adaptive resonance theory. pages 724–729. 17th Mediterranean Conference on Control & Automation, 2009.

[62] M. Yavas and F.N. Alpaslan. Cobart: Correlation based adaptive resonance theory. pages 742–747. 17th Mediterranean Conference on Control & Automation, 2009.

[63] M. Yavas and F.N. Alpaslan. Hierarchical behavior categorization using correlation based adaptive resonance theory. *Neurocomputing (to be published)*, 2011.

APPENDIX A

HIERARCHICAL MODEL EVALUATION USING POSITION DATA

Performance of the hierarchical model for behavior categorization is initially tested by feeding the *CobART Robot* network (see Figure 5.11) with the robot's velocity and position, the *CobART Object* network with the target object's velocity and position, and the *CobART Distance* network with the distance between the robot and the object [63]. Figure A.1 and Figure A.2 show that reasonable categorization is achieved. In the figure, thin black line and black clipped line are used for the robot's velocity and position graphs respectively, light gray clipped line is used for the distance graph, and black circles labeled by numbers are used for the categories.

This categorization process is given in equation (A.1). The *CobART Robot* network of the hierarchical model receives the robot's velocity and position data as an input and generates the robot's self behavior categories as an output. The equation represents the *CobART Robot* network categorization process at time t with c_t^r symbol. In the equation, N, vel^r an d pos^r correspond to the consecutive input number, and the robot velocity and position data, respectively. Similarly, the *CobART Object* network receives the object's velocity and position data as an input and generates the object's self behavior categories as an output. The equation represents the *CobART Object* network categorization process with c_t^o symbol. In the equation, vel^o and pos^o correspond to the object velocity and position data, respectively. Last network of the first layer, the *CobART Distance*, receives the distance between the robot and the object as an input and categorizes it. This categorization is denoted by c_t^d, for time t in the equation, and the distance data is represented as *dist*. The *CobART Relation* network

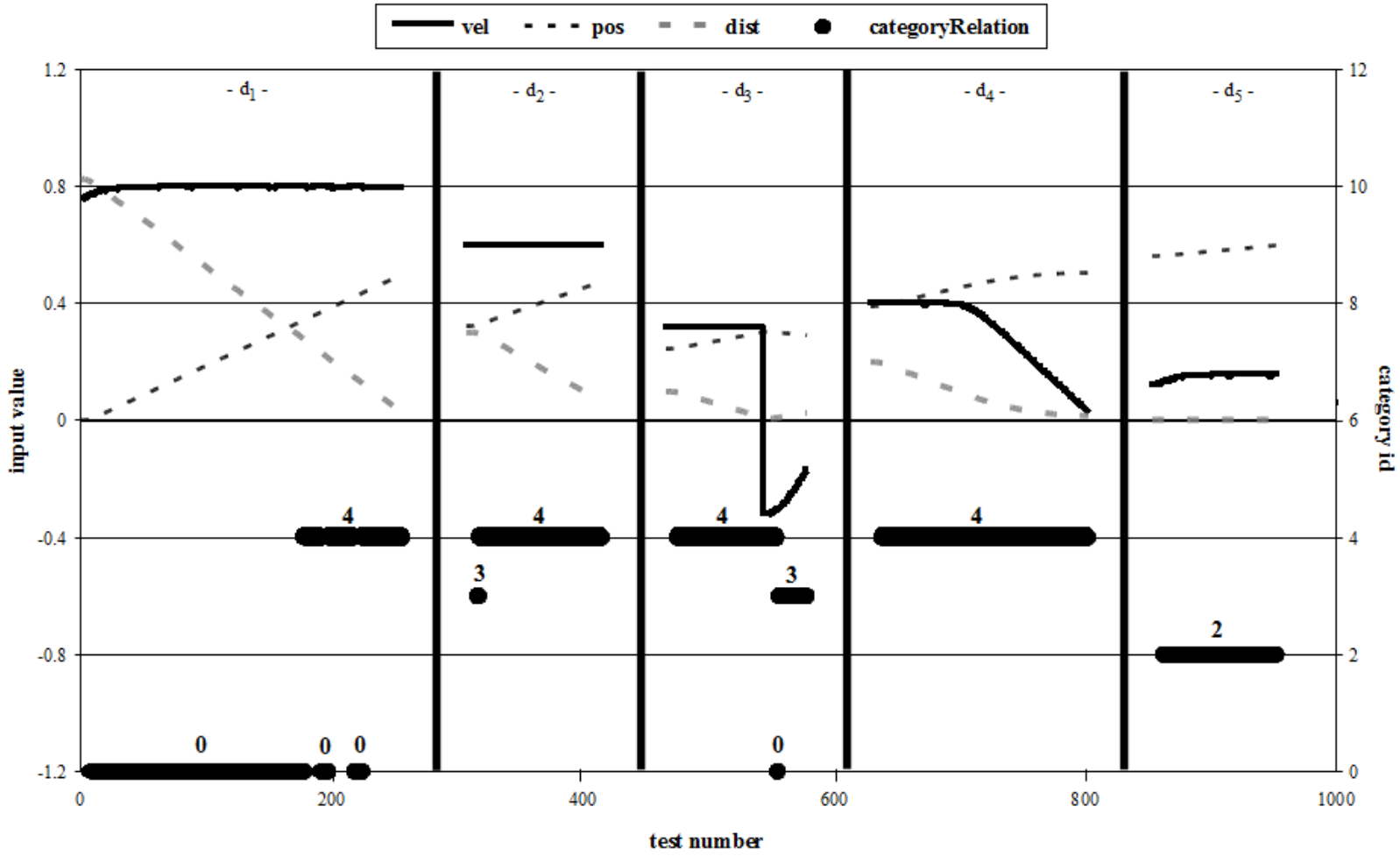

Figure A.1: Behavior categorization by hierarchical model using position data - 1

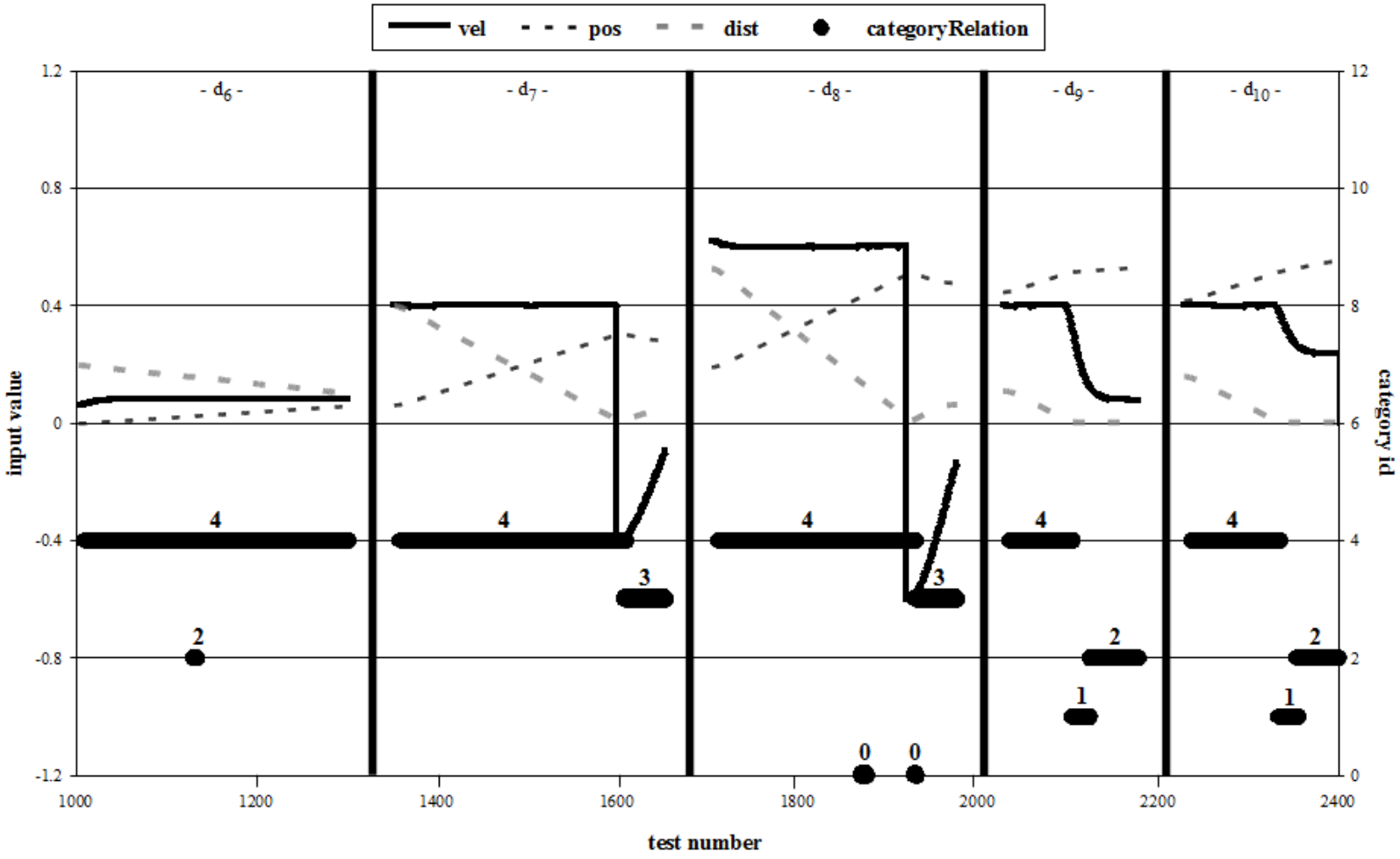

Figure A.2: Behavior categorization by hierarchical model using position data - 2

at the second layer categorizes the robot's behavior with respect to its effect on the target object. Category data from the first layer are fed into the *CobART Relation* network as an input. Hence, categorized robot velocity, robot position, object velocity, object position, and distance data are combined and categorized for the second time. Finally, the *CobART Relation* network categorization for time t is denoted by c_t in the equation.

$$\begin{aligned} c_t^r &= CobART(vel_t^r, .., vel_{t-N+1}^r, pos_t^r, .., pos_{t-N+1}^r) \\ c_t^o &= CobART(vel_t^o, .., vel_{t-N+1}^o, pos_t^o, .., pos_{t-N+1}^o) \\ c_t^d &= CobART(dist_t, .., dist_{t-N+1}) \\ c_t &= CobART(c_t^r, c_t^o, c_t^d) \end{aligned} \tag{A.1}$$

In Figure A.1 and Figure A.2, just the robot velocity, robot position and the distance to the object are given along with the activated categories in order not to complicate the graph [63]. This categorization is obtained when α is -1.5, d is 0.2, a is 0.8, b is 0.7, and N is 10. The vigilance parameter ρ is set to 0.8 for the first layer and 0.6 for the second layer networks. On the other hand, weight parameters w_1, w_2, and w_3 are set to 0.3, 0.3, and 0.4 respectively, in order to make the effect of the *CobART Distance* network higher than effects of the other networks. The feedback weight parameter w_f which will be used to find close correlation between the categories is initialized to 0.2.

Table A.1 summarizes categorization results of the hierarchical model, and presents the mapping between the behaviors and generated categories for each data segment. Figure A.1, Figure A.2 and Table A.1 show that;

- Approaching behaviors are classified with the categories 0 and 4. Category 0 corresponds to an approaching behavior at higher velocity with longer distance when compared to category 4.

Table A.1: *CobART Relation* categories generated for corresponding behaviors when position data is used

data segments	approaching	pushing	striking
d_1	0,4	-	-
d_2	3,4	-	-
d_3 (before 530)	4	-	-
d_3 (after 530)	-	-	3
d_4	4	-	-
d_5	-	2	-
d_6	2,4	-	-
d_7 (before 1580)	4	-	-
d_7 (after 1580)	-	-	3
d_8 (before 1950)	0,4	-	-
d_8 (after 1950)	-	-	3
d_9 (before 2120)	4	-	-
d_9 (after 2120)	-	1,2	-
d_{10} (before 2320)	4	-	-
d_{10} (after 2320)	-	1,2	-

- Pushing behaviors at data segments d_5, d_9, and d_{10} are classified with categories 1 and 2. Category 2 is mapped to a pushing behavior having more stable velocity when compared to category 1.

- Going back after striking behaviors at d_3, d_7, and d_8 are classified with category 3.

According to the test results, only 52 instances of the 1950 test cases are mapped by the wrong categories. This results in 97.3% correctness rate. The significant number of these mistakes are generated during the pass from approaching to pushing or striking behaviors.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Mustafa Yavaş

Nationality: Turkish (TC)

Date and Place of Birth: 6 Feb 1975, Bursa

Marital Status: Single

email: mustafa.yavas@ceng.metu.edu.tr

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Computer Engineering	2001
BS	Hacettepe Uni. Computer Science & Eng.	1998
High School	Bursa Atatürk High School	1992

WORK EXPERIENCE

Year	Place	Enrollment
1998-1999	Hacettepe Uni. Computer Science & Eng.	Research Assistant
1999-Present	ETC Turkey	Software Engineer

FOREIGN LANGUAGES

English

PUBLICATIONS

1. M. Yavas and F.N. Alpaslan. ”Behavior categorization using correlation based adaptive resonance theory”. 17th Mediterranean Conference on Control & Automation, 724-729, 2009.

2. M. Yavas and F.N. Alpaslan. ”Cobart: Correlation based adaptive resonance theory”. 17th Mediterranean Conference on Control & Automation, 742-747, 2009.

3. M. Yavas and F.N. Alpaslan. ”Hierarchical behavior categorization using correlation based adaptive resonance theory”. Neurocomputing (accepted with minor revision), 2011.