Short-Term Memory Mechanisms in Neural Network Learning of Robot Navigation Tasks: A Case Study

Ananda L. Freire*, Guilherme A. Barreto*, Marcus Veloso* and Antonio T. Varela†
*Department of Teleinformatics Engineering, Federal University of Ceará
Av. Mister Hull, S/N - Campus of Pici - Center of Technology, Fortaleza, Ceará, Brazil
Email: anandalf@gmail.com, guilherme@deti.ufc.br, veloso@fisica.ufc.br
†Federal Institute of Education, Science and Tecnology of Ceará
Fortaleza, Ceará, Brazil, Email: themoteo@cefetce.br

***Abstract*—This paper reports results of an investigation on the degree of influence of short-term memory mechanisms on the performance of neural classifiers when applied to robot navigation tasks. In particular, we deal with the well-known strategy of navigating by "wall-following". For this purpose, four standard neural architectures (Logistic Perceptron, Multilayer Perceptron, Mixture of Experts and Elman network) are used to associate different spatiotemporal sensory input patterns with four predetermined action categories. All stages of the experiments - data acquisition, selection and training of the architectures in a simulator and their execution on a real mobile robot - are described. The obtained results suggest that the wall-following task, formulated as a pattern classification problem, is nonlinearly separable, a result that favors the MLP network if no memory of input patters are taken into account. If short-term memory mechanisms are used, then even a linear network is able to perform the same task successfully.**

I. Introduction

This paper reports the results performed with the SCITOS G5 mobile robot on navigation tasks. Basically, using sensor data collected after navigating in a room by following walls, four neural networks architectures are trained to take four decisions that determine its movement. The data are collected by a heuristic classifier based on IF-THEN rules, specifically designed to guide the robot along the walls of a room. Thus, the ultimate goal of the neural classifiers is to mimic the performance of the heuristic classifier as faithfully as possible. The influence of short-term memory (STM) mechanisms on the performance of the classifiers are also investigated in this paper.

The main contribution of this paper comprises (i) the empirical verification that the traditional navigation task via wall-following [1], [2], formulated as a problem of pattern recognition, is a non-linearly separable problem, and(ii) the verification of the hypothesis that STM mechanisms improve the performance of classifiers in the aforementioned task. STM is introduced in two different ways in this work: for static neural networks (e.g. Logistics Perceptrons and MLP), the STM is introduced in the input units through the time-lagged sensory observations, and in the case of dynamic neural networks (e.g. Elman's network), the STM is intrinsic to the architecture itself through recurrent connections.

II. Evaluated Neural Classifiers

Logistic Perceptron - The logistic perceptron (LP) is a single-layered neural network architecture, differing from the simple perceptron network because of the activation function, which is a logistic sigmoid function [3]. Input patterns are represented by the vector $\mathbf{x}(t) \in \mathbb{R}^p$, the weights vector of the i-th output neuron is denoted by $\mathbf{w}_i(t) \in \mathbb{R}^p$, $0 < \eta < 1$ is the rate of learning and $e_i(t) = d_i(t) - y_i(t)$ is the error of i-th neuron for the desired output $d_i(t)$. The matrix transposition is represented by the superscript T, while t denotes the current time step.

The activation and output rules of the i-th neuron are given, respectively, by

$$\begin{aligned} u_i(t) &= \mathbf{w}_i^T(t)\mathbf{x}(t) - \theta_i(t), && (1)\\ y_i(t) &= \phi(u_i(t)) = \frac{1}{1+\exp(-u_i(t))}, && (2) \end{aligned}$$

where $\theta_i(t)$ is the threshold of the i-th neuron and $\phi(u_i(t))$ is the logistic activation function. Thus, the learning rule of the i-th neuron of the LP network is

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e_i(t)\left[y_i(t)(1-y_i(t)\right]\mathbf{x}(t), \qquad (3)$$

where the factor $\phi'(u_i(t)) = y_i(t)(1-y_i(t))$ is the derivative of the activation function with respect to $u_i(t)$.

Multilayer Perceptron (MLP) - In this paper we use the MLP network with only one hidden layer [3]. Both, the hidden and the output layers make use of the logistic sigmoid function. Thus, the output of i-th neuron $y_i(t)$, $i = 1, \ldots, c$, is given by

$$y_i(t) = \phi\left[\sum_{k=1}^{q} m_{ik}(t)\phi\left(\mathbf{w}_k^T(t)\mathbf{x}(t) - \theta_k(t)\right) - \theta_i(t)\right], \quad (4)$$

where $\mathbf{x}(t) \in \mathbb{R}^p$ is the current input vector, $\mathbf{w}_k = [w_{k1}\ w_{k2}\ \cdots\ w_{kp}]^T$ is the weights' vector of k-th hidden neuron, m_{ik} is the weight that connects the k-th hidden neuron to i-th output neuron. The parameters $\theta_k(t)$ and θ_i represent, respectively, the activation thresholds of the k-th hidden neuron and the i-th output neuron.

The weights and thresholds of the output neurons, $\mathbf{m}_i = [m_{i0} \ \ m_{i1} \ \cdots \ m_{iq}]^T$, are adjusted by the rule shown in Eq. (3). Weights and thresholds of the hidden layer are adjusted by classical backpropagation procedure of local errors' gradients of output to the hidden layer.

Mixture of Local Experts - In a sum, the objective of the Mixture of Experts (ME) approach is to model the probability distribution of the training patterns, $\{\mathbf{x}, \mathbf{d}\}$ using a modular architecture with K network experts and a gating network [4], [5]. The gating network determines the output given by the K experts, according to their probabilities g_i, $i = 1, ..., K$, subject to the following constraints: $0 \leq g_i \leq 1$ and $\sum_{i=1}^{K} g_i = 1$. The design of a ME architecture, whose modules consist of LP networks and a single-layered gating network with a *softmax* activation function, is summarized below.

1) *Initialization*: assign small random values to the weights of the experts and gating networks: $\mathbf{w}_i^{(m)}(n)$ and $\mathbf{a}_i(n)$, where $i = 1, 2, ..., K$ denotes the i-th module and $m = 1, 2, ..., c$ denotes the m-th output neuron.
2) *Adapting the Experts and the Gate*: present the current input pattern $\mathbf{x}(t)$ and the corresponding desired response $\mathbf{d}(t)$ to the ME architecture. Then, execute the following operations:

$$u_i(t) = \mathbf{x}^T(t)\mathbf{a}_i(t), \tag{5}$$

$$g_i(t) = \frac{\exp\{u_i(t)\}}{\sum_{j=1}^{K}\exp\{u_j(t)\}}, \tag{6}$$

$$y_i^{(m)}(t) = \mathbf{x}^T(t)\mathbf{w}_i^{(m)}(t), \tag{7}$$

$$\mathbf{y}_i(t) = [y_i^{(1)}(t) \ y_i^{(2)}(t) \ \cdots \ y_i^{(c)}(t)]^T, \tag{8}$$

$$h_i(t) = \frac{g_i(t)\exp\left\{-\frac{1}{2}\|\mathbf{d}(t) - \mathbf{y}_i(t)\|^2\right\}}{\sum_{j=1}^{K} g_j(t)\exp\left\{-\frac{1}{2}\|\mathbf{d}(t) - \mathbf{y}_i(t)\|^2\right\}}, \tag{9}$$

$$e_i^{(m)}(t) = d^m(t) - y_i^{(m)}(t), \tag{10}$$

$$\delta_i^{(m)}(t) = e_i^{(m)}(t)\left[y_i^{(m)}(t)(1 - y_i^{(m)}(t)\right], \tag{11}$$

$$\mathbf{w}_i^{(m)}(t+1) = \mathbf{w}_i^{(m)}(t) + \eta h_i(t)\delta_i^{(m)}(t)\mathbf{x}(t), \tag{12}$$

$$\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \eta\left[h_i(t) - g_i(t)\right]\mathbf{x}(t). \tag{13}$$

3) Repeat step 2 for all training patterns, i.e. $t = 1, 2, ..., N$.
4) Repeat steps 2 and 3 until the process converges.

Elman Recurrent Network - This dynamic neural architecture is obtained from the MLP network by redefining the input layer of the network, which is divided into two parts: the first comprises the usual input vector $\mathbf{x}(t)$, and the second, called units of context, is a copy of the hidden neurons' output at previous time $t - 1$. The weights are adjusted according to the standard backpropagation rule.

III. SCITOS-G5 Mobile Robot

The SCITOS G5 mobile robot, shown in Figure 1, was purchased from MetraLabs Robotics Company

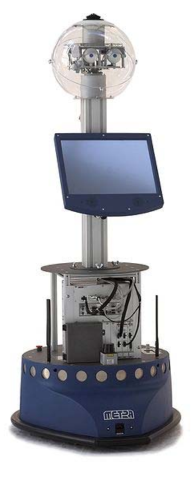

Fig. 1. Robot SCITOS G5.

(http://metralabs.com), located in Ilmenau, Germany. This robot is built for indoors use, moving by means of a differential motor system, carrying its $60kg$ plataform at speeds up to $1.4m/s$, being able to rotate 360° degrees and to push loads up to $50kg$.

The robot has $570mm \times 735mm \times 610mm$ in height, length and width, respectively. With a battery autonomy of $12h$. Its computer hardware consists of a motherboard, Mini ITX, Intel Core Duo Processor 1.6 or 2.0GHz, 2GB of RAM, 120GB of HD, with PS/2 input for keyboard and mouse, 5 USB inputs, 3 Firewire inputs, TV-out, RS232, VGA, SPDIF output, LVDS, and input to RJ-45 Ethernet 10/100 and wireless network on-board, IEEE 802.11a/b/g standard. It uses Linux (Fedora 9) equipped with a collection of libraries C++ API Robot, developed by MetraLabs to enable creation of software for control and navigation of the SCITOS G5 platform. The robot has encoders to calculate the position with 460 ticks by rotation of the wheel, a collision sensor, a belt of 24 ultrasound sensors, with a range within $20cm - 300cm$, a laser sensor SICK S300 and a robotic head with an omnidirectional camera.

IV. Data Collection Procedure

The data collection for training the neural networks has followed the same procedure as [6] and [7]. For this purpose, a routine in C++ was implemented to guide and label the behavior of following walls, given the sensory readings at a given time step. The pseudocode of this heuristic software routine is shown in Algorithm IV.1, which is responsible for generating the decisions that the robot has to take along its navigation and is placed in a thread that is called each time the ultrasound sensors detect any significant change. If there is no sensor activity during $500ms$, the robot simply stops.

Algorithm IV.1: WALL-FOLLOWING($frontDist, leftDist$)

```
if $leftDist > 0,9$
  then
    if $frontDist <= 0,9$
      then Stop and turn to the right
      else Slow down and turn to the left
  else
    if $frontDist <= 0,9$
      then Stop and turn to the right
    else if $leftDist < 0,55$
      then Slow down and turn to the right
    else Move foward
```

The navigation algorithm makes use of heuristic rules IF-THEN, acting on measures of the variables *front distance* and *left distance*, but also calculates the distances to the right and behind. Each of these distances is not calculated by a single sensor, but by a number of them, that together span 60°, and the distance used is the smallest found by those sensors that are part of the set. These four distances are referred throughout this paper as *simplified distances.*

Besides the navigation decision-making, the thread is also responsible for sending, through the wireless network, the following information to the database located at the support computer: 24 ultrasound readings, the (x, y) coordinates and the angular position, the translational and rotational speed of the robot at that time, the label number of the action that was performed according to the settings of the environment and the four simplified distances. This information is collected as the SCITOS G5 navigate through the room, during four rounds. The arrangement of objects located in the test room can be seen in Figure 2 and the trajectory executed by the robot is shown in Figure 3.

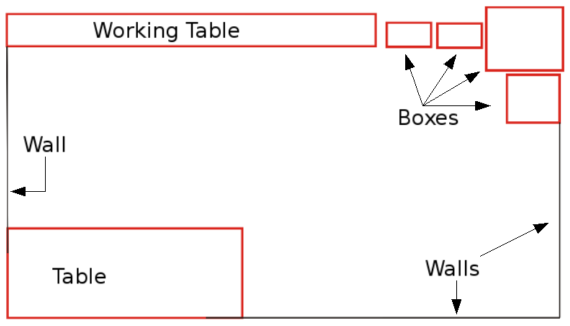

Fig. 2. Sketch of the robot's navigation environment.

The data collection was performed at a rate of 9 samples per second and generated a database with 5456 examples. These data are used to train the neural classifiers presented earlier in order to evaluate which one "mimics" the Algorithm IV.1 better.

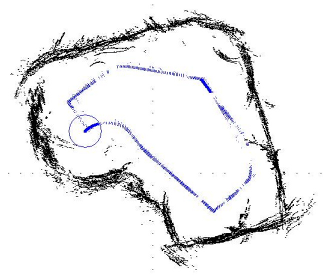

Fig. 3. Perception of the environment and trajectory of SCITOS G5 with Algorithm IV.1.

V. Experiments and Results

Static Classification: The first experiments are performed without the use of STM mechanisms, so that the task of navigation can be formulated as a static classification problem. In this case, the LP network consists of a single layer with $c = 4$ output neurons, each one representing a different action (class): move forward (Class 1), slight right-turn (class 2), sharp right-turn (class 3) and slight left-turn (class 4). The LP network is trained for 200 epochs, the learning rate is set to $\eta = 0.02$. The ME architecture is a network formed by a LP gating network of four neurons, $K = 4$ LP experts, each expert with four output neurons. The training of the ME architecture is carried out for 35 epochs with the learning rate set to $\eta = 0.02$.

For the MLP network, preliminary tests were made with different numbers of hidden neurons to find out the minimum value that enables a suitable class separation. It was found that $q = 6$ is that minimum value. The output layer consists of $c = 4$ neurons with logistics activation function. The MLP is trained for 500 epochs with a learning rate set to $\eta = 0.05$.

For the rest of this paper, we use the following notation: LP (I, O) - logistic perceptron network with I inputs and O output neurons; ME (I, K, O) - mixture of experts network, in which K is the number of experts; MLP (I, q, O) - multilayer perceptron network with q hidden neurons; and Elman($I+q$, q, O) - Elman network with $I+q$ inputs and q hidden neurons.

For the static classification task, we used only 2 input units, corresponding to the simplified distance ahead and to the left of the robot SCITOS-G5. Each experiment was divided into two phases. The first is the offline training/testing on a computer (Pentium 4, 2.4GHz with 500MB of memory, operating system, Ubuntu 8.04), whose results are shown in Tables I and II. This procedure produces an ascii file containing the values of synaptic weights of the trained network. Once this file is available, the resulting neural network architecture can be embedded for robot control purposes, enabling the comparison of its performance with that of the heuristic algorithm.

In the experiment with the network LP(2,4) without STM, the trajectory performed by the robot is shown in

TABLE I
OFFLINE TRAINING STATISTICS WITHOUT STM.

	LP(2,4)	**ME(2,4,4)**	**MLP(2,6,4)**
Average Hits:	3.8%	67.8%	96.82%
Max rate of hits:	13.8%	67.85%	97.95%
Min rate of hits:	0%	67.77%	82.07%
Results Variance:	31311.3	9.69×10^{-8}	9.32×10^{-4}

TABLE II
OFFLINE TESTS RESULTS WITHOUT STM.

	LP(2,4)				**ME(2,4,4)**				**MLP(2,6,4)**			
Mean Squared Error:	0.657797				0.74781				0.0294259			
Success Rate:	42.71%				5.22%				97.59%			
Confusion Matrix:	0	0	594	0	0	602	0	0	571	7	6	0
	0	0	207	0	0	203	0	0	0	213	8	0
	0	0	534	14	0	558	6	0	1	0	559	0
	0	0	19	88	0	85	1	0	11	0	2	78

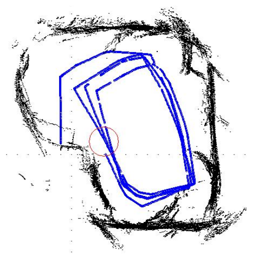
Fig. 4. Perception of the environment and trajectory of SCITOS G5 with LP(2,4) without STM.

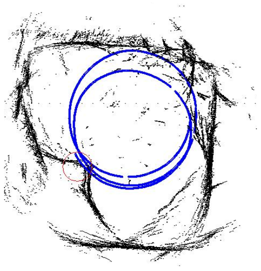
Fig. 5. Perception of the environment and trajectory of SCITOS G5 with ME(2,4,4) without STM.

Figure 4. One can observe that there was no collision, but the robot was not able to execute the left-turn. A reduced discrimination ability was expected because of the low number of samples correctly classified in the corresponding confusion matrix (Table II). With the architecture ME(2,4,4) without STM, the robot was able to almost complete a round in the room before colliding (Figure 5). The robot was repositioned from the local it collided, but when it was supposed to turn left, he turned right and remained in that movement. Analyzing the actions stored in the database, we noted that the robot used, mostly, the action *slight right-turn* and, in a very few attempts, a sharp right-turn. The MLP network achieved the best performance, as shown in Figure 6. Its recognition rates are shown in Table II.

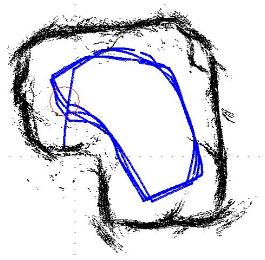
Fig. 6. Perception of the environment and trajectory of SCITOS G5 with MLP(2,6,4) without STM.

Spatiotemporal Classification: Once the robot behavior is assessed on a static classification task (i.e. without STM), additional experiments are conducted to evaluate (1) if the use of STM improves the performance of the classifiers that performed poorly in the previous experiments, and (2) if the use of STM allows a reduction in the number of hidden neurons in the MLP architecture, thus reducing its computational cost.

The first network to be tested is the LP architecture, which is fed not only with the current simplified distances, measured by the ultrasound sensors, but also by their recent past values. Except for the change in the dimensionality of the input vector, the training parameters remain the same as the experiments conducted without STM. The LP network was tested with $1, 2, 3, 5, 9, 34$ and 49 past samples (observations) of the simplified distances. Only the results with 9 past samples of these distances are presented in this paper, since it was the best configuration found. Thus, the input vector is composed by the current values and the past nine values of each of the two simplified distances, resulting in an input vector with 20 components. Analyzing the results shown in Figure 7 and Tables III and IV, we find a clear improvement in the performance of the LP network with the inclusion of STM mechanisms.

For the ME architecture, the use of time-lagged observations did not improved the performance of the robot navigation at all and hence the results are not shown in

TABLE III
Offline training statistics with STM.

	LP(20,4)	Elman(2+4,4,4)
Average Hits:	13.57%	85.16%
Max rate of hits:	14.7%	96.42%
Min rate of hits:	13.55%	75.52%
Results Variance:	0.43	0.00390116

TABLE IV
Offline tests results with STM.

	LP(20,4)				Elman(2+4,4,4)			
Mean Square Error:	0.232588				0.0555338			
Success Rate:	67.08%				96.22%			
Confusion Matrix:	287	135	146	19	578	9	7	0
	93	110	4	0	10	197	0	0
	10	0	522	13	6	3	539	0
	47	0	9	51	17	0	3	87

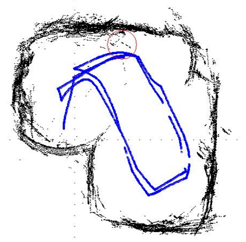
Fig. 7. Perception of the environment and trajectory of SCITOS G5 with LP(2,4) with STM.

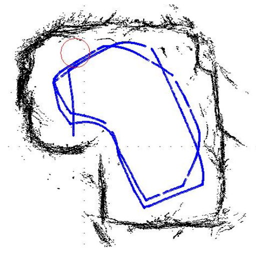
Fig. 8. Perception of the environment and trajectory of SCITOS G5 with Elman(2+4,4,4).

this paper.

An alternative STM mechanism can be implemented through recorrent connections [8]. To determine whether this type of STM improves the performance of multilayered neural architectures, the Elman network was evaluated in the robot navigation task. As input, the Elman network uses only the current readings of ultrasound sensors together with q context units. The training parameters are the same the MLP network used for the static classification task, except the rate of learning ($\eta = 0.05$). After several tests varying the number of hidden neurons, it was found that the minimum value for the robot to successfully perform the required task is $q = 4$, 2 less neurons than the MLP network used in static classification problem. It is important to emphasize that the network MLP(2,6,4) was already able to solve the task without STM; however, the inclusion of STM through recurrent loops as in Elman network, allowed the optimal classifier to have less hidden neurons.

Decision Surfaces: A method to evaluate the performances of the classifiers in the navigation task of interest is through visualization of the decision regions implemented by each one. In Figures 9(a) and 9(b) one can note that the inclusion of STM allowed a better separation of classes by the classifiers, in the case of LP network. Each point in figures corresponds to a pair of measurements of simplified distances in a given moment of time along the trajectory executed by the robot. The colors help differentiating the sensory readings belonging to each of the four classes defined for the navigation task (class 1 - red; class 2 - green; class 3 - blue; class 4 - yellow).

In short, we find an improvement in the distinction of four classes by LP network with STM if we compare with LP without STM. The MLP and Elman's networks, which achieved the best performances in the navigation task, produced similar decision surfaces (Figures 9(c) and 9(d)), although the Elman network achieved its result with two neurons less in the hidden layer.

VI. Conclusion

This study compared the performance of four neural classifiers used as controllers in the wall-following navigation task in a real mobile robot. For this, the navigation task is formulated as a pattern recognition problem, in which the patterns are the sensor readings and classes are actions to be taken by the robot. We evaluated the logistic perceptron network, the mixture of experts architecture (with experts trained with logistic perceptron networks), the multilayer perceptron network and the Elman recurrent network. In general, the MLP network achieved the best performance among the four classifiers evaluated.

The experiments showed that an apparently simple navigation task is indeed a complex decision-making task. The use of short-term memory mechanisms by the logistic perceptron classifier allowed it to successfully accomplish

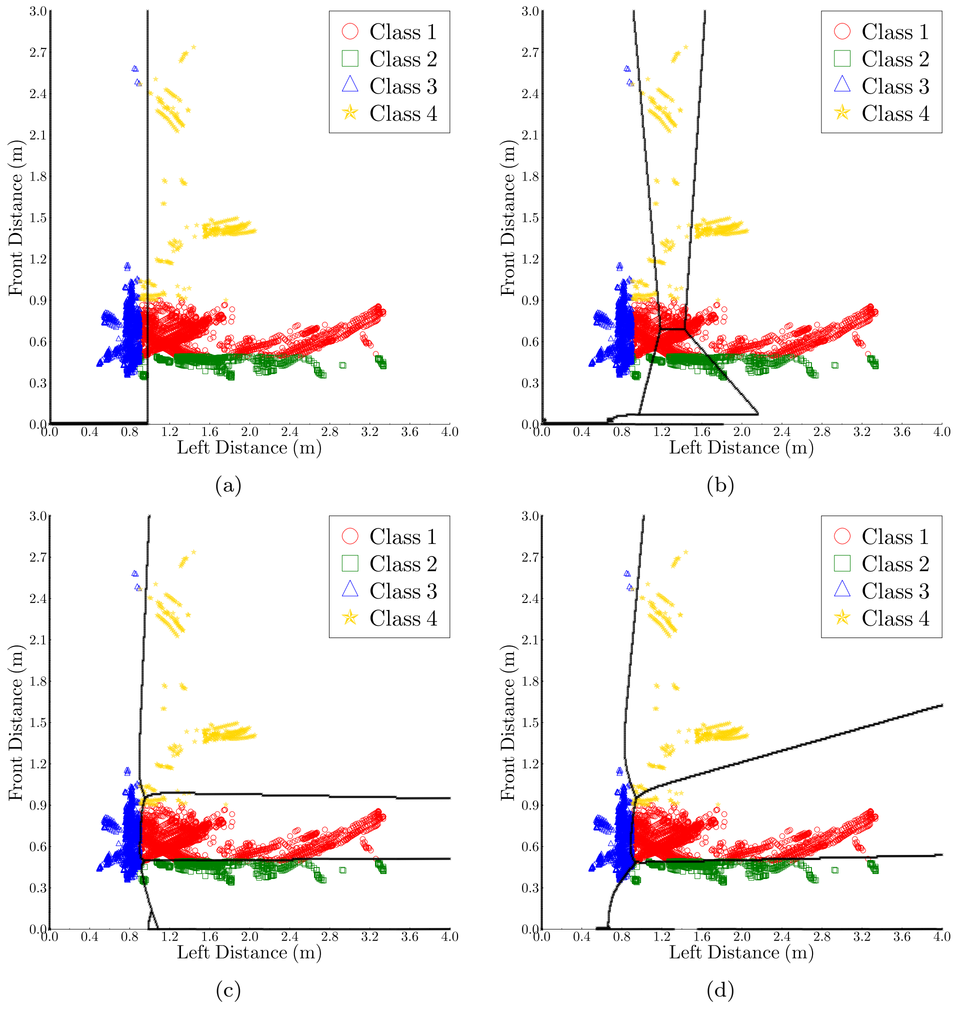

Fig. 9. Decision Surfaces of a) LP(2,4) without STM, b) LP(20,4), c) MLP(2,6,4) and d) Elman(2+4,4,4).

a nonlinear separable classification task. The use of short-term memory also allows, via recurrent loops, a multilayered neural architecture to reduce the number of hidden neurons, without compromising the discriminative ability of the classifier.

In future works, we aim at investigating the performance of SVM classifiers on the wall-following navigation task as well as evaluating other types of short-term memory mechanisms, such as the one implemented by the *Echo-state network* recurrent architecture [6].

Acknowledgments

The authors would like to thank FUNCAP and SECITECE for financial support.

References

[1] G. A. Bekey, *Autonomous Robots*. MIT Press, 2005.
[2] K. M. Krishna and P. K. Kalra, "Spatial understanding and temporal correlation for a mobile robot," *Spatial Cognition and Computation*, vol. 2, no. 3, pp. 219–259, 2000.
[3] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1994.
[4] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, no. 3, pp. 79–87, 1991.
[5] E. Alpaydin and M. I. Jordan, "Local linear perceptrons for classification," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 788–792, 1996.
[6] E. Antonelo, B. Schrauwen, and D. Stroobandt, "Mobile robot control in the road sign problem using reservoir computing networks," in *IEEE International Conference on Robotics and Automation (ICRA'2008)*, 2008, pp. 911 – 916.
[7] M. Mucientes, R. Alcalá, and J. Alcalá-Fdez, J.and Casillas, "Learning weighted linguistic rules to control an autonomous robot," *International Journal of Intelligent Systems*, vol. 24, no. 3, pp. 226–251, 2009.
[8] G. A. Barreto, A. F. R. Araújo, and S. C. Kremer, "A taxonomy for spatiotemporal connectionist networks revisited: The unsupervised case," *Neural Computation*, vol. 15, pp. 1255–1320, 2003.