A generic neural network for multi-modal sensorimotor learning

F. Carenzi, P. Bendahan, V.Y. Roschin*, A.A. Frolov*, P. Gorce, M.A. Maier

INSERM U.483, University Pierre et Marie Curie, 9 quai Saint Bernard, Paris, France

*Institute of Higher Nervous Activity, Moscow, Russia

Phone: +33-1- 44 27 34 21, Fax: +33-1- 44 27 34 38, E-mail: Marc.Maier@snv.jussieu.fr

Abstract

A generic neural network module has been developed, which learns to combine multi-modal sensory information to produce adequate motor commands. The module learns in a first step to combine multi-modal sensory information, based on which it subsequently learns to control a kinematic arm. The module can learn to combine two sensory inputs whatever their modality. We report the architecture and learning strategy of the module and characterize its performance by simulations in two situations of reaching by a linear arm with multiple degrees-of-freedom: 1. mapping of tactile and arm-related proprioceptive information, and 2. mapping of gaze and arm-related proprioceptive information.

Keywords: Hebbian learning, multi-network architecture, multi-modal sensory information, sensorimotor integration.

1. Introduction.

There has been a resurgence of interest in artificial neural networks for the control of movements over the last few years. In particular, neural networks have been used as controllers for simulated or robotic arms, as well as for representing internal models of the kinematic or dynamic properties of the arm [2]. In addition, some of these neural networks had been derived from neuroscientific perspective, either in terms of architecture or in terms of learning rules: Schweighofer et al [5] have implemented a neural network model of the cerebellum for the control of a simulated dynamic arm. Recently, the acquisition of an internal representation of the motor or the somato-motor system by the use of neural networks has been further illustrated [3], [1].

During natural movements, the central nervous system (CNS) receives sensory information of various modalities, however, how afferent information is put into register with the motor command and how afferent information is used to control and modify a motor command is not fully understood. Various neural network models have been developed to approach these questions [e.g. 4]. Within this framework, we present the architecture and the learning procedure of a generic neural network module for the treatment and use of multi-modal sensory input, exemplified by two cases: first, a mapping between tactile and proprioceptive information, second a mapping between oculo- and skeleto-motor proprioceptive signals for the learning and control of arm movements. The module is capable of learning the relation between two sensory inputs of different modality and their respective relation to the motor command. The architecture and learning rule is invariant with respect to the particular combination of modalities. This generic neural network module thus learns to combine multi-modal sensory information to produce adequate kinematic reaching commands.

2. Architecture of the neural network

The goal is to develop a neural network architecture of a generic module capable of learning multi-modal sensorimotor relations independently of the specific nature of the sensory signals. The generic module (Fig. 1) consists of two sub-networks: a generic "matching unit" and a generic "motor command unit". The matching unit correlates two sensory signals from different modalities and learns through minimizing the mismatch between those two signals. The motor command unit receives inputs from the matching unit and provides a motor command in the kinematics domain. These two sub-networks use Hebbian learning rules. Learning is performed sequentially, first in the matching unit, then in the motor command unit. Here we present two identical matching units, each of which treats a different combination of sensory signals. The first concerns the learning of the relation between a tactile and a proprioceptive signal, whereas the second learns the relation between two proprioceptive signals (one oculo-motor, the other skeleto-motor). The matching unit learns to map the two signals, each of which codes in its own modality the 3D positional information of the end-point of the arm, into a common space. The motor command unit then learns the transformation from the initial position of the arm to the target position, as defined by the error vector. Since the originality of our approach is based on the concept of the matching unit, we have simplified the motor command unit: it drives a linear arm with 4 degrees-of-freedom (DoF) represented by the Jacobian matrix (*J*), which transforms an angular configuration of the arm (*Q*) into a cartesian 3D end-point (*X*). The inverse transformation is computed as follows:

$$Q = pinv(\boldsymbol{J}) * X + (\boldsymbol{Id} - pinv(\boldsymbol{J}) * \boldsymbol{J}) * random \qquad (1)$$

$pinv(\boldsymbol{J}) * X$ represents a particular solution and $(\boldsymbol{Id} - pinv(\boldsymbol{J}) * \boldsymbol{J}) * random$ the homogenous solution, $pinv(\boldsymbol{J})$ is the pseudoinverse of ***J***, and ***Id*** is the identity matrix.

Figure 1 about here

2.1 Simulator of sensory signals

The sensory simulator provides coding of tactile and proprioceptive signals. Input vector1 (*V1*) represents the angular information of the articular chain. Vector2 (*V2*) defines the target, given either by tactile or by oculomotor proprioceptive (gaze) information. The size of the proprioceptive vector is defined by the number of DoF of the arm, the size of vector *V2* depends on whether it codes tactile or oculomotor information, but is different from *V1*.

The neural coding of inputs *V1* and *V2* is performed with the matrices *M1* and *M2*, that provide the neurally coded sensory signals *A1* and *A2* of identical dimensionality, i.e. the inputs to the generic matching unit (Fig. 1)

$$A1 = \boldsymbol{M1}.V1 \qquad (2)$$

$$A2 = \boldsymbol{M2}.V2 \qquad (3)$$

2.2 Simulator of Neural Network

The generic neural network is divided into two sub-networks: a matching unit and a motor-command unit.

2.2.1 Generic matching unit

In a first learning step, the matching unit correlates the two sensory signals *A1* and *A2* (of different modality). The matrices *Mr1* and *Mr2*, initially randomized, map the two inputs into a common representation, i.e. in one case, proprioceptive and tactile information, in the other case, oculomotor and skeletomotor proprioceptive information. Thus, the two informations can now be expressed in a common space, independently of their modality:

$$Ar1 = \boldsymbol{Mr1}.A1 \qquad (4)$$

$$Ar2 = \boldsymbol{Mr2}.A2 \tag{5}$$

The learning procedure adapts the matrices so that the two signals converge. The learning signal is obtained from the difference Ar between these vectors:

$$Ar = Ar1 \square Ar2 \tag{6}$$

The learning procedure modifies the scalars (weights) in *Mr1* and *Mr2* in order to minimize the difference Ar. The Hebbian learning rule is given by:

$$\boldsymbol{Mr1} = \boldsymbol{Mr1} + k.Lr1.Ar.A1 \tag{7}$$

$$\boldsymbol{Mr2} = \boldsymbol{Mr2} \square Lr2.Ar.A2 \tag{8}$$

where K and Lr (learning rate) are constants. After learning, given two independent sensory inputs *V1* and *V2*, Ar represents the difference between them in a common space.

The 2nd learning step generates an approximate motor command *Amu*. This step relates a given command to its sensory consequences. We use an initial and random command *Amu* to generate the corresponding sensory signal *V1*, which is injected into the matching unit. Having established the matching matrices during the first learning step, *V1* now generates the output *Ar1*. Given *Ar1*, Hebbian learning adapts *U* so that the new motor command *Amu'* tends toward *Amu*.

2.2.2 Motor command unit

The goal of the motor command unit is to transform the neural and approximate command *Amu* into a motor command *Amv* via matrix *V*. In the 3rd learning step, *V* learns to minimize the number of iterations from the initial to the target state. *V* is modified by Hebbian learning. Based on *Amv*, the matrix *Mm* computes the angular configuration of the linear simulated arm. The new angular configuration then provides the input to the simulator of the sensory signals and closes the loop.

3. Simulation results

3.1 Simulation for tactile-proprioceptive learning

The first simulation concerns the learning of the relation between tactile and proprioceptive signals. Target position is defined in the tactile and arm position in the proprioceptive domain. This corresponds to a situation where the endpoint of the arm should touch a particular part of the body. The tactile map consisted of a spherical surface composed of 20 tactile receptive fields. The tactile target position is defined by a receptive field index and a 2D vector in the plane of the receptive field, coding the tactile target point within the field with respect to its center.

Through learning, a correlation is obtained between the angular configuration of the arm and the target on the sphere, coded in tactile space. The method consists in mapping a tactile-derived target position to the corresponding proprioceptive configuration of the arm when its endpoint contacts the target point. This is akin to a procedure called motor-babbling. Fig. 2 shows the asymptotic learning curve for Ar which tends to zero, indicating an increasingly better correlation between the two sensory signals.

In the 2nd learning step, an *approximate* motor command is generated through learning in matrix *U*. The matrix *U* learns the proprioceptive consequences of a random motor command. The learning curve of *U* is shown in FIG. 3.

FIG 2 and 3 about here

In the third learning step, by learning in matrix *V*, we generate a motor command from an initial (randomly chosen) position to a randomly chosen target position on the sphere. Now that the multi-modal sensory signals are correlated in terms of coding a common point in

3D space, we can quantify the error *Ar* when the joint angles do not coincide with the target position, i.e. in the initial situation for target reaching. This error serves to generate the motor command *Amv*.

At the end of this learning phase, we obtain a correct angular configuration of the arm in a single time step, after having specified a target position. FIG. 4 presents the asymptotic learning curve of *V*: its values tend to zero, indicating that learning in *V* does indeed predict the correct angular configuration to reach the target.

FIG 4 about here

3.2 Simulation for the learning of gaze and proprioceptive signals

The second simulation uses gaze (oculomotor proprioception) for the definition of the target and, as above, arm proprioception.

The oculomotor coding is based on the 3 angles of gaze (azimuth, elevation, and vergence), which determine a 3D point in the working space of the arm. The actual working space is, in this case, restricted to a cube (representing 25 % of the full working space) divided into 10 vertical slices or subspaces (analogue to the way the tactile surface was divided into multiple receptive fields). The aim of the learning is thus to provide a code in a common space of gaze and arm proprioception, and to determine the angular configuration of the arm in order to reach a target coded by the 3 angles of gaze. The method and sequence of learning are identical to the previous case, but the modality of the sensory signals is different.

FIG. 5 shows the learning curves for *Ar*, the difference between the coding of the two sensory inputs. Compared to the tactile-proprioceptive situation, the learning is slower and the error larger by 2 orders of magnitude. FIG. 6 shows the learning curve for *U*, the approximate motor command.

FIG. 5 and 6 about here

Finally, the motor command that leads in a single step from a random initial position to the target position defined by gaze is learned in matrix V. After learning, the correct angular configuration of the arm is obtained in a single time step. FIG. 7 shows that the learning in V is less accurate and saturates faster than in the tactile–proprioceptive case.

FIG. 7 – about here

We have thus demonstrated the iterative convergence of the three subsequent learning procedures: i) the correlation between two multi-modal sensory signals, ii) the generation of an approximate and iii) the generation of a target motor command. This has been simulated for a linear arm in two cases: in the tactile-proprioceptive domain and in the oculo- and skeleto-motor proprioceptive domain.

4. Conclusions

This paper presents a novel learning architecture based on the association of two generic neural networks. This neural network has been tested in a simulated learning environment of a 4 DoF linear arm. Learning occurs in three distinct and sequential epochs: the first is dedicated to the correlation (match) between multi-modal sensory information by the use of a common coding space, the second is based on motor babbling and associates the sensory consequences of random movements, and the third then learns, based on the previous two, to move from an initial to a target position in terms of kinematics. The mapping between the two sensory spaces is achieved via the division of the workspace into subspaces, either by tactile receptive fields or by oculomotor subspaces in depths. The future development concerns three aspects: first, the introduction of a non-linear arm including multi-joint interactions, second, a more realistic coding of sensory signals, and third, the implementation of a neurobiologically inspired architecture based on interactions of multiple matching and motor command units.

Acknowledgements

This work was in part supported by RFBR grant 01-04-48924a and a European 5th framework programme grant: IST-2001-33073.

References

[1] M. Haruno, D. Wolpert, M. Kawato, Hierarchical MOSAIC for movement generation, in: T. Ono, G. Matsumoto, RR Llinas, A. Berthoz, R. Norgren, H. Nishijo, R. Tamura, eds., Excepta Medica International Coungress Series, *1250*, (Elsevier Science B.V. in press)

[2] M. Kawato, Robotics as a tool for neuroscience: cerebellar internal models for robotics and cognition, in: M.H. Hollerbach, D.E. Koditschek, eds., Robotics Research, The Ninth International Symposium 2000*,* London, pp.321-328.

[3] H. Miyamoto, D.M. Wolpert, M. Kawato, Computing the optimal trajectory of arm movement: the TOPS (Task Optimization in the Presence of Signal-dependent noise) model. RJ Duro, J. Santos, M. Grana, eds., (Studies in Fuzziness and Soft Computing 2000) 395-415.

[4] E. Salinas, L.F. Abbott, Transfer of coded information from sensory to motor networks. J Neurosci 15 (1995) 6461-6474.

[5] N. Schweighofer, M.A. Arbib, M. Kawato, Role of cerebellum in reaching movement in humans. I. Distributed inverse dynamics control. Eur J Neurosci, 10 (1998) 86-94.

Figure captions:

FIG. 1. The general architecture contains a generic neural network module linked to a simulator of sensory signals (SS) and to a linear model of the arm. The module is made up of a "matching unit" and a "motor command unit".

FIG. 2 (left). Tactile-proprioceptive match. Learning curves of □Ar □for each of the 20 receptive fields; 300 000 iterations; log y-scale.

FIG. 3 (right). Tactile-proprioceptive learning. Learning curve for the approximate motor command (180000 iterations).

FIG. 4. Tactile-proprioceptive learning. Learning curves for V (target motor command, 5500 iterations for each of the 20 receptive fields).

FIG. 5 (left). Match of gaze and arm proprioception. Learning curves of □Ar □; 300000 iterations.

FIG. 6 (right). Match of gaze and arm proprioception. Learning curve for the approximate motor command (120000 iterations).

FIG. 7. Match of gaze and arm proprioception. Learning curves for V (target motor command, 6000 iterations for each of the 10 subspaces).

Fig. 1

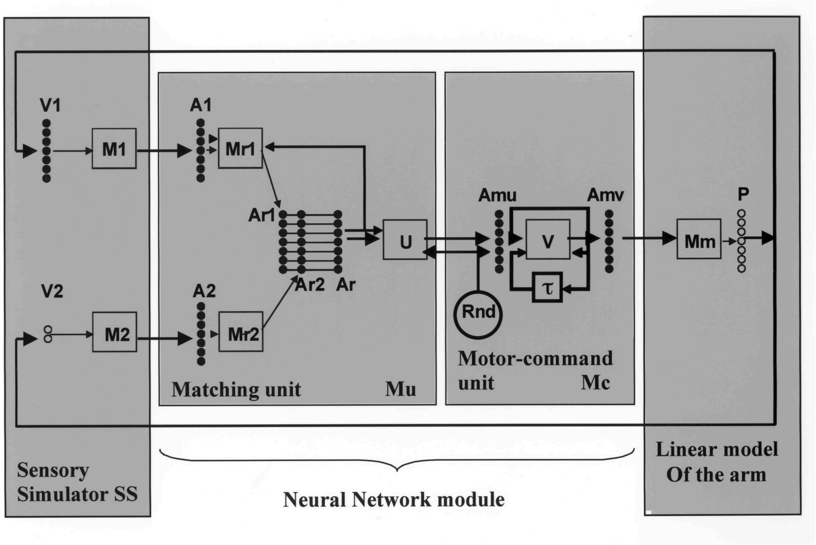

V1
M1
A1
Mr1
Ar1
U
Ar2
Ar
V2
M2
A2
Mr2
Amu
Amv
V
τ
Rnd
P
Mm
Matching unit
Mu
Motor-command unit
Mc
Sensory Simulator SS
Neural Network module
Linear model Of the arm

Fig. 2

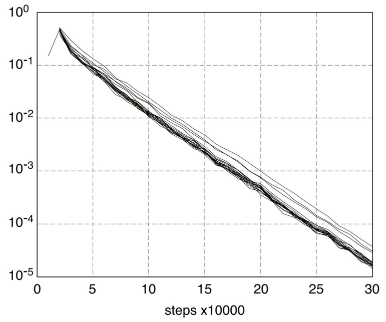


Fig. 3

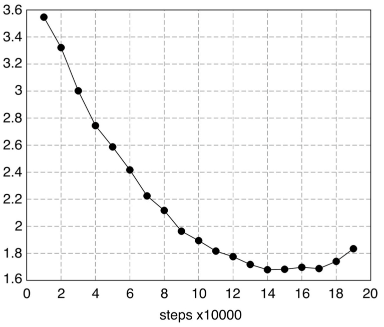


Fig. 4

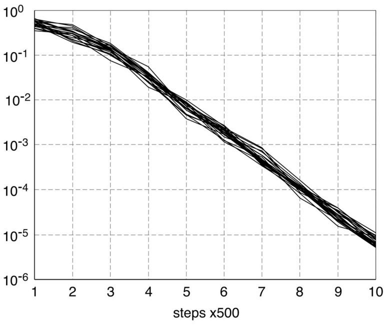


Fig. 5

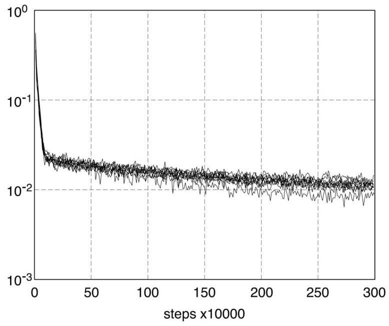


Fig. 6

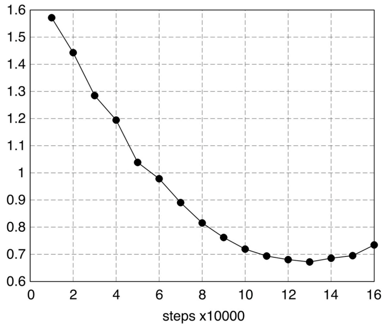
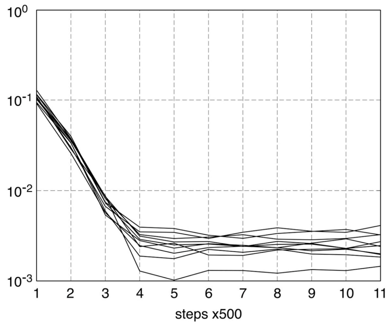


Fig. 7


10^{0}
10^{-1}
10^{-2}
10^{-3}
steps x500