Sensorimotor Learning

Lina L. E. Massone
Dept. of Electrical Engineering and Computer Science
Dept. of Biomedical Engineering
Northwestern University
2145 Sheridan Road, Evanston, Il 60208

RUNNING HEAD: Sensorimotor Learning

Correspondence:
Prof. Lina L.E. Massone
MEAS/EECS, Room Tech 1573
Northwestern University
2145 Sheridan Road, Evanston, Il 60208-3118
Phone: (708) 491-7297
Fax: (708) 491-4455
Email: *massone@eecs.nwu.edu*

To appear in "The Handbook of Brain Theory and Neural Networks", Michael A. Arbib Editor, MIT press

1. INTRODUCTION

A sensorimotor transformation maps signals from various sensory modalities into an appropriate set of efferent motor commands to skeletal muscles or to robotic actuators. Sensorimotor learning refers to the process of tuning the internal parameters of the various structures of the central nervous system (CNS) or of some processing architecture in such a way that a satisfactory motor performance will result.

Sensorimotor transformations and sensorimotor learning can be viewed as complex parallel distributed information processing tasks. A number of different components contribute to the tasks' complexity. On the sensory side, signals from various sources (vision, hearing, touch, pain, proprioception) need to be integrated and interpreted (Stein and Meredith, 1993). In particular, they need to be translated into a form that can be used for motor purposes because the coordinates of afferent sensory signals are different from the coordinates of the movements they are guiding. This convergent process, from many parallel sensory signals to an intermediate internal representation, is referred to as the early stages in a sensorimotor transformation. A widely accepted concept that describes the computed internal representation is the so-called motor program: a specification of the important parameters of the movement to be executed (Keele and Summers, 1976). Bernstein used the term motor program to denote a prototype of a planned movement described with an abstract central language that encodes specific features of the movement itself. This idea was then elaborated by Schmidt (1988) and modified by Arbib (1990), who introduced the concept of coordinated control program: a combination of perceptual and motor functional units called schemas, whose dynamic interaction causes the emergence of behavior. A different approach was proposed by Schoner and Kelso (1988), who redefined motor programs in dynamic terms as transitions between the equilibrium states of a dynamical system. Interestingly, there is some experimental evidence that some of the parameters of the motor programs could be actuator-independent, i.e. unrelated to the particular plant that will carry out the movements: Bernstein (1967), for example, observed in his classic experiments the presence of similar patterns in cursive writing executed with different joints and/or with different parts of the body. Motor programs, however, are likely to be task-dependent entities and to exhibit different structures and different coordinate frames for different classes of movements, like, for example, reaching, grasping, manipulating.

A quite different process is needed in order to translate the computed internal representations into a set of commands to the actuators (muscles/motors). This process is referred to as the late stages in a sensorimotor transformation. An important feature of the late stages is their intrinsic divergence: it is a mapping from a lower to a higher number of degrees of freedom. There are in fact many more muscles in, let's say, an arm than there are independent variables in a representation of movement expressed, let's say, in a body-centered coordinate frame. The redundancy of the motor system is an important and desirable property: it allows the same motion to be carried out with different muscle combinations under different task-related and environmental circumstances. Hence it provides the basis for the organism to be able to cope, for example, with fatigue and fast reaction. But, most importantly, redundancy underlies the smoothness of body movements: without a redundant system we would be moving like puppets. It is important to understand that redundancy is exploited by the CNS in order to coordinate body movements efficiently. Consequently, any model of sensorimotor tranformations and sensorimotor learning should be based on the same constructive approach - exploit redundancy in order to enrich the motor repertoire - rather than try to eliminate redundancy – a destructive approach.

One additional degree of complexity in the transformation from sensory to motor is the fact that such transformations occur continuously over time but with basic events that occur at various different time scales and various different spatial resolutions, and hence require different grains of parallelism (Arbib, 1993). Moreover, not only the information needs to be transformed between different coordinate frames, but often a translation is required from signals that are spatially coded to signals that are temporally coded and vice versa – the so-called spatio-temporal and temporal-spatial transformations. Interesting examples of such transformations can be found in the saccadic/gaze system (Massone, in press; Massone and Khoshaba, submitted).

How is this tremendous complexity handled by the CNS and how does the CNS learn what the correct architectural and processing choices are for the large ensemble of neurons involved? What are the important aspects that need to be taken into account when trying to model such a complex process?

2. COMPUTATIONAL APPROACHES

From a computational perspective, it is important to point out that many aspects of sensorimotor transformations and learning can be captured by the network-like style of computation of connectionist models. While the correspondence between such a computational style and brain mechanisms remains an open and controversial issue, it is true that some of the basic properties of connectionist models match very well the problems that one must face when dealing with sensorimotor transformations and learning.

2.1 Coordinate Transformation

The transformation f that occurs between an n-dimensional input layer and an m-dimensional output layer of a network

$$\underline{y} = f(\underline{x}), f : R^n \longrightarrow R^m \tag{1}$$

can be viewed as a very general, nonlinear coordinate transformation process. This view becomes particularly important when one uses it in association with error correction mechanisms and learning algorithms. The ability of connectionist learning algorithms to solve the credit assignment problem constitutes an efficient mechanism for translating errors expressed in one particular coordinate frame (R^m) into errors expressed in a different coordinate frame (R^n). The relevance of this approach for motor control purposes lies in the fact that often errors are available, through some measuring system, only in task or task-related coordinates and not in the coordinate frame of the controller. One example related to arm movements is that of a hand-coordinate error measured by the visual system, that needs to be translated into an error in muscle-activation space – error that is not directly available. Another example involving a different system, the vestibulo-ocular reflex, is the translation of the retinal slip signal measured by the accessory optic system into gain and phase errors. These two examples, although quite different in nature, share a common general structure, based on two primary components: (a) A controlled object that translates a motor command into an actual movement, i.e. into task coordinates; (b) A controller that generates the motor commands for the controlled object. The computation of the motor commands and their translation into movements (task coordinates) is referred to as the forward path. The opposite computation, from task coordinates to motor commands, is referred to as the inverse path. In particular, the inverse path is what is needed in order to learn an appropriate controller for a given controlled object. The next subsection will introduce three different approaches that have been proposed in the literature for this particular learning task. A more detailed account of this topic can be found, for example, in Jordan and Rumelhart (1992), Kawato (1990). See also CONTROL THEORY, BIOLOGICAL.

2.2 Forward and Inverse Models

A forward model is a representation of the transformation from motor commands to movements, in other words a model of the controlled object. An inverse model is a representation of the transformation from desired movements to motor commands. Hence an inverse model can be used as the controller for the controlled object: the cascade of the two implements an identity mapping as shown in Figure 1. How can the inverse model of a controlled object be learned?

A first method is the so-called direct inverse modeling. The approach (see Figure 2) consists of activating the controlled object (or a forward model of it) with some motor commands generated, for example, in a random fashion. The output produced by the controlled object is used as the input to the inverse model, that in turn computes an estimated motor command. The estimated motor command is then compared to the actual motor command and an error is computed. The error can be used to modify the inverse model. If the inverse model is, for example, a multilayer network with randomly initialized weights, the error can be used to update the network's weights with supervised learning techniques. If this process (generation of random motor command, error computation, weights update) is repeated long enough, the network will learn, over time, to behave like the inverse of the controlled object. The trained network can now be used as a controller as shown in Figure 1. This learning scheme, although conceptually simple, has, in practice, a number of drawbacks as pointed out by Jordan (1990), most notably the fact that it cannot properly control nonlinear redundant objects (i.e. nonlinear objects with a lower number of degrees of freedom in task space than in motor command space) because it cannot solve for the inherent ambiguity of the inverse transformation. It is worth pointing out that the inability of handling redundancy is a very serious drawback, because redundancy is, as emphasized in the introductory section, an intrinsic, extensive and important property of the musculoskeletal system.

A different and more effective approach is the distal supervised learning approach of Jordan and Rumelhart (1992) that consists of two phases depicted in Figure 3. During the first phase (see Figure 3a) the forward model is learned by activating the controlled object and the forward model with the same motor command, again randomly generated. The error between the output of the forward model and the output of the controlled object is then computed and used to update the weights of the network that represents the forward model. Over a number of learning trials the forward model learns to emulate the controlled object. The second phase consists of training the controller. This is achieved with the scheme shown in Figure 3b: A desired trajectory is input to the controller, the actual movement is computed at the output of the controlled object and an error between the two is computed. The error is then propagated backwards through the forward model and transformed into a motor command error, which is in turn used to update the weights of the controller. As learning progresses, the controller becomes an inverse model of the controlled object. The two phases can also be merged, so that the forward model and the controller can be learned in parallel. The forward model approach presents significant advantages over direct inverse modeling. First of all, because of its goal-oriented character, it can properly cope with nolinear redundant objects. In addition, it can be used simultaneously for learning and for control, while direct inverse modeling can't. Another interesting feature of the forward model approach is that the same scheme can be used to learn an input-output transformation that is not necessarily an identity mapping. This would be most useful, for example, in the case of a controller that transforms a sensory stimulus or an internal representation of it into motor commands, as in the late stages of a sensorimotor transformation.

A third approach, called feedback-error-learning and proposed by Kawato (1990), is depicted in Figure 4. In that scheme an error is computed between a desired trajectory and the actual trajectory executed by the plant. A so-called feedback controller, which is a linear approximation of the inverse model, transforms the trajectory error into a motor-command error, which is used to train the inverse model and to compute the actual motor command that activates the plant. The latter is computed as the sum of the motor-command error plus the motor command computed by the inverse model. As learning progresses and the error decreases the feedback controller play a lesser and lesser role in the computation until the plant is completely controlled by its inverse model. Like the forward model approach, feedback-error-learning is able to control successfully nonlinear redundant plants. For a biological interpretation of feedback-error-learning see CEREBELLUM AND MOTOR CONTROL.

2.3 Optimization and Learning

A computational approach to error transformation based on distributed learning schemes must be able to deal with the redundancy present at the various stages of sensorimotor transformations. Because the pathway for the transformation of a motor program expressed in task coordinates into patterns of muscle activation is a divergent one, there is redundancy in the inverse tranformation from an error expressed in task coordinates to an error expressed in muscle coordinates: There are infinite possible combinations of muscle errors that would lead to the same task error. In order to solve for the redundancy problem in a way that is meaningful from the standpoint of the tasks being carried out, one must make use of optimization techniques so that the error transformation process will make specific choices for the solutions, rather than arbitrary ones. One way of tying optimization to learning is to incorporate optimization principles in the learning algorithms themselves, i.e. to constrain the learning procedure to follow specific paths. Most connectionist learning algorithms are formulated as the minimization of a cost function. Consequently, one possible approach to learning with extra degrees of freedom is to include in the cost function additional terms (besides the usual quadratic error terms) that represent optimization criteria in some analytical form.

Let's consider, for example, a model of the late stages of the sensorimotor transformation of a planar reaching movement executed by a 3-joint, 8-muscle arm. Let's assume (see Figure 5) that in such a model the controller maps an internal representation of the movement target into a time-varying pattern of activity of the eight muscles of the arm. The controlled object is the arm itself or a model of its biomechanical properties; it transforms the activation of the muscles into joint angles (forward dynamics) and then the joint angles into cartesian coordinates of the hand (forward kinematic). In this model the number of degrees of freedom in muscle space is higher than the number of degrees of freedom in joint space, which is in turn higher than the number of degrees of freedom in hand space. Hence the controlled object presents both a kinematic and a dynamic redundancy: there is an infinite number of ways in which an error in hand coordinates maps into an error in joint coordinates, and an infinite number of ways in which an error in joint coordinates maps into an error in muscle coordinates. Let's now adopt the forward model approach to learning the controller, which requires to propagate the hand error backwards in order to compute an error in muscle space – a highly ill-posed process in our example. Let's then look at the cost function J that the learning algorithm minimizes. If we choose J as a simple quadratic form of the hand error

$$J = \frac{1}{2}(\underline{x}^* - \underline{x})^T(\underline{x}^* - \underline{x}) \tag{2}$$

with corresponding learning rule for the network weights $\underline{w}$

$$\Delta\underline{w} = -\alpha\nabla_{\underline{w}}J \tag{3}$$

$$\nabla_{\underline{w}}J = -(\underline{x}^* - \underline{x})^T\frac{\partial\underline{x}}{\partial\underline{w}} \tag{4}$$

then the resulting error in muscle space will be picked at random among the infinite possible configurations – a solution that is unlikely to bear any physiological significance. The cost function can however be enriched to include terms that force the learning process to compute the error in muscle space according to a specific criterion. The following cost function, for example, realizes an optimization criterion of minimum change of the muscles activation (see also Kawato 1992). It is the sum of the quadratic cost function (2) plus a term that penalizes the change over time of the vector $\underline{m}$.

$$J = \frac{1}{2}(\underline{x}^* - \underline{x})^T(\underline{x}^* - \underline{x}) + \frac{1}{2}\sum_{k=1}^{N}(\underline{m}_k - \underline{m}_{k-1})^T(\underline{m}_k - \underline{m}_{k-1}) \tag{5}$$

In the second part of the equation k represents the time index and N the number of time steps in the reaching movement. The corresponding learning rule is given by

$$\Delta \underline{w} = -\alpha \nabla_{\underline{w}} J \tag{6}$$

$$\nabla_{\underline{w}} J = -(\underline{x}^* - \underline{x})^T \frac{\partial \underline{x}}{\partial \underline{w}} + \sum_{k=1}^{N} (\underline{m}_k - \underline{m}_{k-1})^T \frac{\partial \underline{m}_k}{\partial \underline{w}} - (\underline{m}_k - \underline{m}_{k-1})^T \frac{\partial \underline{m}_{k-1}}{\partial \underline{w}} \tag{7}$$

By substituting $\underline{q}$ for $\underline{m}$ in equations (5) to (7) one can obtain a learning rule that solves for the redundancy of the arm by minimizing the change over time of the joint angles, i.e. the angular velocity.

The use of modified cost functions to learn with extra degrees of freedom was first introduced by Jordan (1990) in the context of arm movements and speech: he introduced smoothness constraints (of which equation (5) is an example) as well as their opposite, distinctiveness contraints, and rest-configuration constraints. Alternative and successful ways of optimizing the learning process have been proposed by Kawato (1992). Massone and Bizzi (1989) utilized a minimum-potential-energy criterion to construct off-line the training set for an arm controller. Flash and Hogan (1985) and Uno et al. (1989) constitute, respectively, the first kinematic and dynamic optimization models based on data from human subjects.

2.4 Representing Information

Another aspect that plays a major role in the computation underlying sensorimotor learning is how the information is represented. It is known that a design principle widely used by the CNS is that of coarse coding (Hinton et al, 1986): large and partially overlapped receptive fields. Through this principle the CNS achieves greater sensitivity and resolution, higher signal-to-noise ratio, wider dynamic range, better resistance to damages due to loss or independent drift of single neurons, reduced demand for precision in ontogenesis and for stability of performance. Coarse coding turns out to be also crucial in the design of computational systems capable of learning sensorimotor mappings. Massone and Bizzi (1990) provided a comparison between the performances of a learning algorithm for an arm controller when different types of representation were used for the sensory information at the network input layer. The comparison showed that a coarse coding scheme is superior, in terms of both number of epochs of the learning algorithm and the ability of the controller to generalize, to other more local schemes. The reason for the latter effect is that a coarse-coding scheme requires a lower number of units to represent a certain amount of information than a local scheme does and hence the corresponding network exhibits a lower number of computational degrees of freedom. The reason why coarse coding is also likely to require a lower number of epochs to learn a certain set of patterns is because the receptive fields of the input units overlap. This causes an input unit to become active more often than in a local scheme. Consequently, the weights that originate from each input unit are trained, in the case of coarse coding, more often than in the case of local coding.

Acknowledgements.

This work was supported by the National Science Foundation, grant n. BCS-9113455 to Lina L.E. Massone. Thanks to Michael Arbib, Michael Jordan and Mitsuo Kawato for their comments on early drafts of this paper.

REFERENCES

Arbib, M.A., 1990, Programs, schemas and neural networks for control of hand movements: beyond the RS framework, in Attention and Performance XIII, (M. Jeannerod Ed.), Hillsdale:Lawrence Erlbaum, pp. 111-138.

Arbib, M.A., 1993, What Frogs' Brain Tell Robots' Schemas, in Neural Networks in Robotics, (G.A. Beckey and K.Y. Goldberg Eds.), Boston:Kluwer Academic Publishers, pp. 503-519.

* Bernstein, N.A., 1967, The Coordination and Regulation of Movements, Oxford:Pergamon Press.

Flash, T., and Hogan, N. 1985, The coordination of arm movements: an experimentally confirmed mathematical model, Journal of Neuroscience, 5:1688-1703.

Hinton, G.E., McClelland, J.L., and Rumelhart D.E., 1986, Distributed Representations, in Parallel Distributed Processing 1, (D.E. Rumelhart, and J.L. McClelland Eds.), Cambridge:MIT Press, pp. 77-109.

* Jordan, M.I., 1990, Motor Learning and the Degrees of Freedom Problem, in Attention and Performance XIII, (M. Jeannerod Ed.), Hillsdale:Lawrence Erlbaum, pp. 796-836.

Jordan, M.I., Flash, T., and Arnon, Y., in press, A Model of the Learning of Arm Trajectories from Spatial Deviations, The Journal of Cognitive Neuroscience.

Jordan, M.I., and Rumelhart, D.E., 1992, Forward Models: Supervised Learning with a Distal Teacher, Cognitive Science, 16:307-354.

Kawato, M., 1990, Computational Schemes and Neural Network Models for Formation and Control of Multijoint Arm Trajectory, in Neural Networks for Control, (W.T. Miller, R.S. Sutton, P.J. Werbos Eds.), Cambridge:MIT Press, pp. 197-228.

* Kawato, M., 1992 Optimization and Learning in Neural Networks for Formation and Control of Coordinated Movement, in Attention and Performance XIV, (D. Meyer Ed.), Hillsdale:Lawrence Erlbaum, pp. 821-849.

* Keele, S.W., and Summers, J.J., 1976, The Structure of Motor Programs, in Motor Control - Issues and Trends, (G.E. Stelmach Ed.), San Diego:Academic Press, pp. 109-142.

Massone, L., and Bizzi, E., 1989, A Neural Network Model for Limb Trajectory Formation, Biological Cybernetics, 61:417-425.

Massone, L., and Bizzi, E., 1990, On the Role of Input Representations in Sensorimotor Mapping, Proc. IJCNN, Washington D.C., 1:173-176.

Massone, L., in press, A Neural Network System for Control of Eye Movements: Basic Mechanisms, Biological Cybernetics.

Massone, L., and Khoshaba, T., submitted, Local Dynamic Interactions in the Collicular Motor Map: A Neural Network Model, Network.

* Schmidt, R.A., 1988, Motor Control and Learning: A Behavioral Emphasis, Champaign:Human Kinetics.

Schoener, G., and Kelso, J.A.S., 1988, Dynamic Pattern Generation in Behavioral and Neural Systems, Science, 239:1513-1520.

* Stein, B.E., and Meredith, M.A., 1993, The Merging of the Senses, Cambridge:MIT Press.

Uno, Y., Kawato, M., Suzuki, R., 1989, Formation and control of optimal trajectory in human multijoint arm movement: minimum-torque-change model, Biological Cybernetics, 61:89-101.

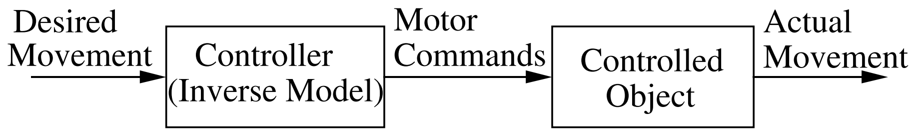

Figure 1: An identity mapping realized as the cascade of an inverse model and the corresponding controlled object.

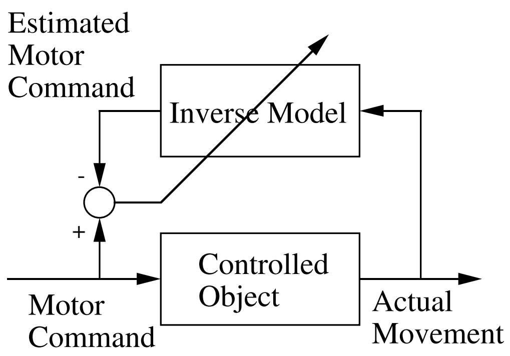

Figure 2: Direct inverse modeling.

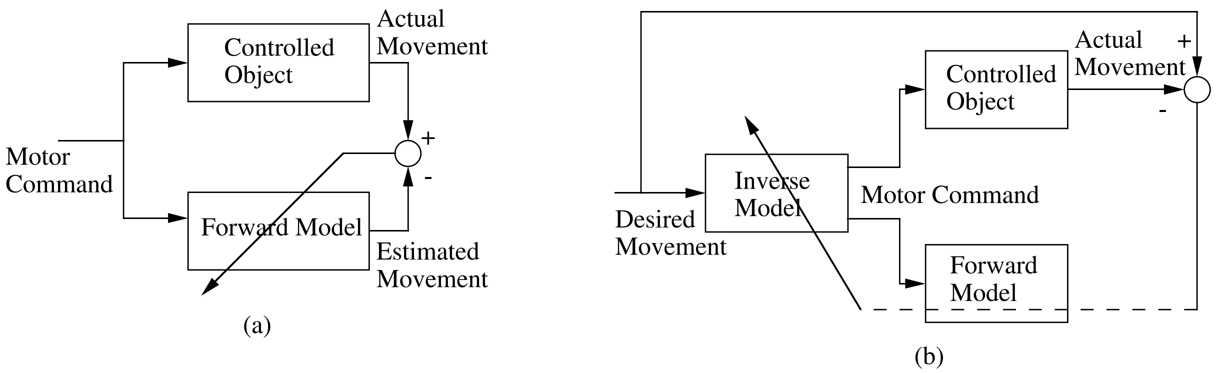

Figure 3: (a) Learning the forward model. (b) Learning the controller.

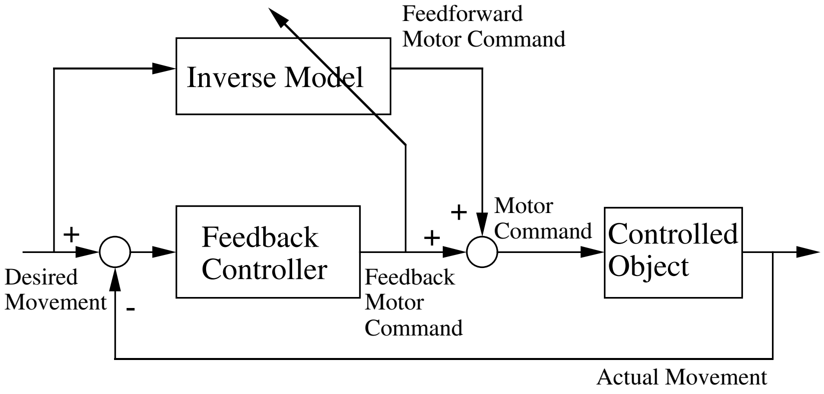

Figure 4: Feedback-error-learning.

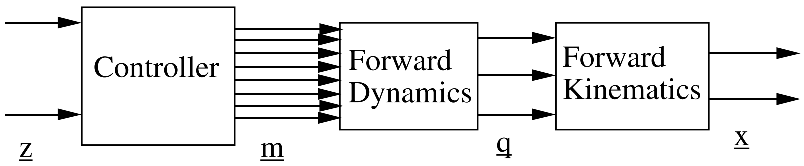

Figure 5: A sensorimotor transformation from an internal representation of a sensory stimulus ($\underline{z}$) to hand coordinates ($\underline{x}$); $\underline{m}$: muscle activations, $\underline{q}$: joint angles.