ELSEVIER

Neurocomputing 37 (2001) 31–49

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Polyhedral mixture of linear experts for many-to-one mapping inversion and multiple controllers

Amir Karniel[a,*,1], Ron Meir[b], Gideon F. Inbar[b]

[a]*Department of Physiology, Northwestern University Medical School, 303 East Chicago Avenue, Chicago, Illinois 60611, USA*

[b]*Department of electrical engineering, Technion Israel Institute of Technology, Haifa 32000, Israel*

Received 12 June 1999; accepted 13 April 2000

Abstract

Feed-forward control schemes require an inverse mapping of the controlled system. In adaptive systems this inverse mapping is learned from examples. The biological motor control is very redundant, as are many robotic systems, therefore the mapping is many-to-one and the inverse problem is ill posed. In this paper we present a novel architecture and algorithms for the approximation and inversion of many-to-one functions. The proposed architecture retains all the possible solutions available to the controller in real time. This is done by a modified mixture of experts architecture, where each expert is linear and more than a single expert may be assigned to the same input region. The learning is implemented by the hinging hyperplanes algorithm. The proposed architecture is described and its operation is illustrated for some simple cases. Finally, the virtue of redundancy and its exploitation by multiple controllers are discussed.

Keywords: Redundancy; Motor control; Inverse problem; Mixture of experts; Hinging hyperplanes

1. Introduction

One of the salient characteristics of the biological motor control system is its apparent redundancy (see e.g. [1,18]). The human arm consists of seven degrees of

* Corresponding author.

E-mail address: karniel@nwu.edu (A. Karniel).

[1] This study was done while Dr. Karniel was at the Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel.

0925-2312/01/$ - see front matter
PII: S0925-2312(00)00306-4

freedom, which is more than needed to obtain a particular position or configuration of the hand in the workspace. Most of the joints are surrounded by more muscles than needed to produce any desired moment. The muscles themselves are composed of many motor units that enable many possibilities of producing the same force at the tendon. In the presence of redundancy the controller has to act on a many-to-one (MTO) system and has to choose one of the many possible actions to obtain the same desired target.

Due to the delays in the nervous system, simple feedback cannot offer a proper explanation for the control of fast movement. Thus, it was suggested that the nervous system contains an inverse model of the musculo-skeletal system that is contextually being updated [7]. For reviews of recent modeling with artificial neural networks, see [10,14]. Two methods for learning this inverse model are distal supervised learning [11] and feedback error learning [16]. These methods do not confront the MTO problem. They choose an arbitrary solution that is the closest to the training set and to the initial conditions of the network. In some cases these architectures may incorporate a smoothness criterion to choose a biologically plausible solution, but they still learn just one solution.

The same MTO problem occurs in the robot inverse kinematics problem for manipulators with excess degrees-of-freedom. This problem can be separated into global and local ill-posedness, and therefore a two-fold solution can be pursued: first global regularization, that is identifying and labeling the solution branches and then local regularization corresponding to parameterization of the solution manifolds. DeMers [4] investigated this problem by describing the topological properties of the systems. He suggested such a two-fold learning method. Lu and Ito [20] tried to solve the inverse kinematics of a redundant arm with a modular neural network, where each network learned part of the configuration space; but as they admit, the regions can overlap.

In this paper a simpler and more tractable and analyzable method is suggested. The main idea is to construct a piecewise invertiable approximation that can be then inverted to produce a multiple controller. We present a novel architecture that divides the input space into polyhedral regions, which are convex regions that can cover the whole space. We call this special architecture polyhedral mixture of linear experts (PMLE) because it can be viewed as a special case of the mixture of experts (ME) architecture proposed by Jacobs and Jordan [8]. The PMLE has the advantage of being a piecewise invertiable function and therefore the multiple inverse PMLE can serve as a multiple controller in order to exploit the virtue of redundancy. We use the hinging hyperplanes (HH) method proposed by Breiman [3] in order to learn the piecewise linear approximation. Then we present a new algorithm to transform the parameters of the HH to the parameters of the PMLE. We further prove the ability of the PMLE to estimate inverse functions. Part of this work was presented as a short conference paper [12] and further details are available in Chapters 5 and 6 of [15].

The reminder of the paper is organized as follows: Section 2 describes the inverse problem and its ill-posedness. In the next two sections the HH algorithm and the PMLE architecture are described. The PMLE is shown to be capable of approxim-

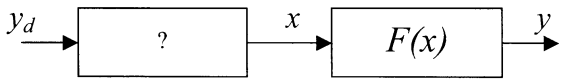


Fig. 1. The inverse problem: given a desired output y_d, find x such that $F(x) = y_d$.

ating any inverse function. In Section 5 the transformation and inversion algorithms are outlined and in Section 6, a few examples of its performance are given. Finally, the role of the PMLE in utilizing the virtue of redundancy in biological and artificial systems is briefly discussed and conclusions are drawn.

2. Learning to invert many-to-one mappings

The problem of finding an inverse mapping is described in Fig. 1. Given a desired output y_d, find x such that $F(x) = y_d$. For example, given the desired position of the hand, what should be the neural excitations to the muscles in order to bring the hand to the desired position. In a robotic system, the question is what should be the currents in each motor, in order to bring the manipulator to the desired position. When such a problem is given many questions can be formalized, for example: (I) Is there a solution? (II) Is the solution unique? (III) Is there an algorithm to find all the solutions? (IV) If there is more than one solution, which of them is optimal?

In this work a system with many solutions is considered and a solution to the third question is proposed. The problem gets more complicated when the system mapping $F(X)$ is unknown or uncertain. Then the inverse mapping should be learned from examples of input and outputs pairs $\{x^l, y^l\}$. This description is appropriate for biological motor control learning, where the system and the environment changes and therefore have to be learned from examples. It is also appropriate for robotics applications where the manipulator is too complex to be modeled accurately, or when its properties changes over time.

After this introduction we can write the formal description of the problem and the proposed solution, which are given below and illustrated in Fig. 2. Let $F(x)$ be a many to one function describing an unknown system, and let $\{x^l, y^l\}$be a series of input and output vectors of this system. The problem is to construct a multiple inverse function $\hat{F}_{\mathrm{P}}^{\mathrm{MI}}(y_d)$, where MI stands for multiple inverse and the parameter p determines which of the many possible solutions is chosen. The formal requirement is that for any given accuracy value ε, one can construct $\hat{F}_{\mathrm{P}}^{\mathrm{MI}}$, so that for any value of y_d, and for any value of the parameter p, the following inequality will hold, $|F(\hat{F}_{\mathrm{P}}^{\mathrm{MI}}(y_d)) - y_d| < \varepsilon$, that is, the output of the system will be close to the desired output.

In the following sections we will describe the hinging hyperplanes as the learning algorithm and the PMLE architecture which can serve as a multiple inverse controller.

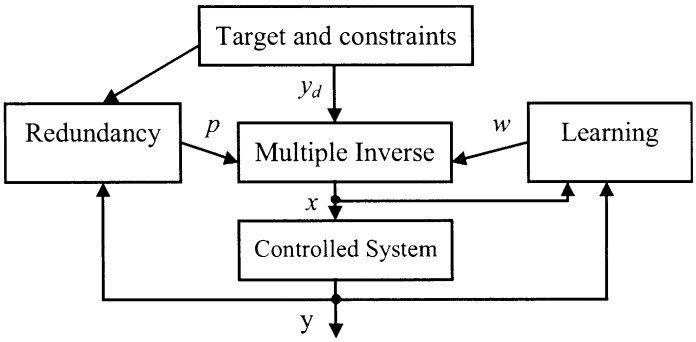


Fig. 2. The proposed scheme for learning the inverse control of a many-to-one system. The multiple inverse controller receives the desired target y_d. Its parameters w are learned from examples and it produces the control command, x, according to a given criterion to choose one of the many possible solutions parameterized by p.

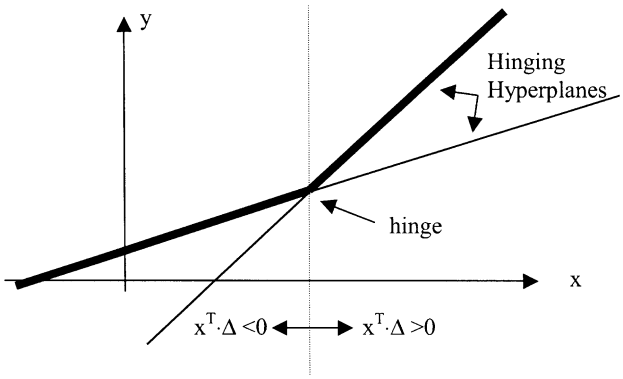


Fig. 3. Hinge, hinging hyperplanes and hinge function (bold) in one dimension).

3. Hinging hyperplanes

The problem of finding a model for an unknown system by observing a set of input–output examples has many solutions in the adaptive control and neural networks literature (see [23] for a unified overview). In order to be able to invert the approximate system, an estimation with a kernel of an invertible function is needed. A linear function is an appropriate one and the hinging hyperplanes (HH) method proposed by Breiman [3] is an elegant and efficient way of identifying piecewise linear models based on data collected from an unknown system. In this section the HH method is described following the definitions of Breiman [3], and some further investigations by Pucar and Sjöberg [22].

A hinge function $y = h(x)$ consists of two hyperplanes continuously joined together at a hinge. An example in one dimension is given in Fig. 3.

In an M-dimensional space, taking $x_0 = 1$, that is $x = [1, x_1, \ldots, x_M]^T$, the two hyperplanes are given by $y = x^T\beta^+$ and $y = x^T\beta^-$, and are joined together on $\{x|x^T(\beta^+ - \beta^-) = 0\}$. The vector, $\Delta \equiv \beta^+ - \beta^-$, or any multiple of Δ, is defined as the *hinge* of the two hyperplanes. The hinge can be described geometrically as the $(M-1)$-dimensional hyperplane that satisfies $x^T \cdot \Delta = 0$. The explicit form of the hinge function is either $\max(x^T\beta^+, x^T\beta^-)$ or $\min(x^T\beta^+, x^T\beta^-)$. In this paper, we define $\Delta \equiv \beta^+ - \beta^-$, when the function is max, and, $\Delta \equiv \beta^- - \beta^+$, when the function is min. In this way the hinge function is always:

$$h(x) = \begin{cases} x^T\beta^+, & x^T \cdot \Delta \geq 0, \\ x^T\beta^-, & x^T \cdot \Delta < 0. \end{cases}$$

Given data from an unknown function, one can fit a hinge to the data by the following algorithm, due to Breiman [3].

3.1. The hinge finding algorithm (*HFA*)

Start with an arbitrary hinge $\Delta^{(0)}$. Using least squares, fit the data on the side $x^T \cdot \Delta^{(0)} \geq 0$ to a hyperplane $y = x^T\beta^+$, and the data on the other side $x^T \cdot \Delta^{(0)} < 0$ to a hyperplane $y = x^T\beta^-$. Take the new estimate for the hinge as $\Delta^{(1)} = \beta^+ - \beta^-$, and repeat this procedure until the error is small enough. Breiman [3] proved that the convergence of this algorithm is exponentially fast when the data is taken from an unknown hinge function and when the initial hinge is close to the target hinge. This simple algorithm was used successfully in the simulations of Section 6, however, there is place for further improvement of this algorithm. Pucar and Sjöberg [22] illustrated situations where the algorithm does not converge and suggested modified algorithms.

3.2. Function approximation by hinging hyperplanes

Given data from an unknown function $f(x)$ one can construct an approximation of this function as a sum of hinge functions.

$$\hat{f}(x) = \sum_{k=1}^{K} h_k(x), \quad h_k(x) = \begin{cases} x^T\beta_k^+, & x^T \cdot \Delta_k \geq 0, \\ x^T\beta_k^-, & x^T \cdot \Delta_k < 0. \end{cases} \tag{1}$$

Breiman [3] proved that for a continuous and sufficiently smooth function the squared approximation error decreases as one over the number of hinge functions that are used for the approximation. Several algorithms for fitting the hinge functions were proposed by Breiman [3] and by Pucar and Sjöberg [20]. Following is the basic refitting method:

1. Find $h_1(x)$ estimation of $f(x)$ by the HFA, set $K = 2$.
2. Find $h_K(x)$ estimation of $f(x) - \sum_{k=1}^{K-1} h_k(x)$ by the HFA.

3. Refit by the HFA:

$$h_1(x) \Rightarrow f(x) - \sum_{k=2}^{K} h_k(x)$$
$$\vdots$$
$$h_i(x) \Rightarrow f(x) - \sum_{k=1,k\neq i}^{K} h_k(x)$$
$$\vdots$$
$$h_K(x) \Rightarrow f(x) - \sum_{k=1,}^{K-1} h_k(x).$$

4. Set $K = K + 1$, repeat 2–4 until convergence

(i.e. until residual sum of squares (RSS) ceases to decrease significantly)

The stopping condition can be improved in order to prevent overfitting. One possible way is based on cross-validation, which means keeping a portion of the data out of the training set and using it in order to calculate the RSS. Another way can be to add a stopping condition when the number of data points in each area is too small. We definitely do not want the number of points in each area to be of the order of the dimension of the input due to the problem of over-fitting.

4. Polyhedral mixture of linear experts

4.1. The architecture

In the mixture of experts (ME) architecture of Jacobs et al. [9] the input is fed to a group of experts, and the output is a weighted sum of the experts' output. These weights are also a function of the input through the gate (see Fig. 4 and Eq. (2)).

$$y = \sum_i g_i(x, \theta) \cdot f_i(x, w), \quad \sum_i g_i(x, \theta) = 1, \quad g_i(x, \theta) \geq 0. \tag{2}$$

The polyhedral mixture of linear experts (PMLE) is a special case of the ME architecture where each expert is a linear function, that is a weighted sum of the inputs, and the gate function is an indicator function that separates the input space into a polyhedral partition and assigns to each polyhedron a unique linear expert, as follows:

$$f_i(x, w) = x^{\mathrm{T}} w, \quad g_i(x, \theta) \in \begin{cases} 1 & \text{if } x \in \text{ polyhedron } i, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

where $x = [1, x_1, \ldots, x_{\mathrm{M}}]^{\mathrm{T}}$, $w = [w_0, w_1, \ldots, w_{\mathrm{M}}]^{\mathrm{T}}$

A polyhedron is a subspace of R^n composed of the intersection of a finite number of half-spaces. The polyhedral experts can easily cover the whole input space and each polyhedral region is convex. This architecture is actually an implementation of a piecewise linear mapping, capable of approximating inverse functions, as will be proved in the next subsection. Then an algorithm to transform the HH parameters into this architecture will be given, so that the HH algorithm may be used to learn the parameters of the PMLE from input/output examples.

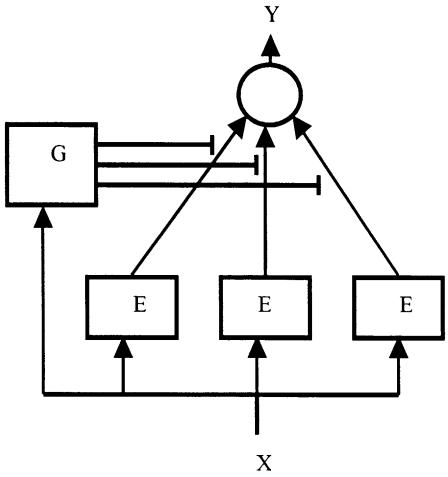


Fig. 4. The mixture of experts architecture: each expert (E) computes a function of the input. The gate (G) chooses a weighted sum of the experts output to be the output of the system.

4.2. The ability to approximate inverse functions

The ability to approximate inverse functions was studied by Sontag [24]. He showed that neural networks with a single hidden layer are insufficient, but that two hidden layer networks are able to approximate any inverse map. In this section, it will be shown that the PMLE is also able to approximate any inverse map, and actually able to approximate any function that can be approximated by a two hidden layer neural network with sharp threshold activation functions (H activation function). An activation function of type H is the hardlimiter, Heaviside or threshold function, i.e. $H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x \geq 0$. First, let us recall the definition of the ability to approximate inverse functions from Sontag [24]. The following is a property of the class of functions F_p^m, from R^p to R^m.

(INV) For any m and p, any continuous function $f: R^m \to R^p$, any compact subset $C \subseteq R^p$ included in the image of f, and any $\varepsilon > 0$, there exist some $\Phi \in F_p^m$ so that $\| f(\Phi(x)) - x \| < \varepsilon$ for all $x \in C$.

Theorem. *The class of functions that are computable by the* PMLE *satisfies* (INV), i.e., *any inverse function can be approximate by the* PMLE.

Proof. First let us recall the following proposition and lemma from [24]:

Proposition 2.4 (Sontag [24]). F_p^m, *the set of maps computable by two-hidden-layer nets with processors of type* H, *satisfies* (INV).

Lemma 3.6 (Sontag [24]). *A function f is piecewise constant if and only if it is computable by a two-hidden-layer net with processors of type H.*

Now all we have to add is the trivial observation that a piecewise constant function is a special case of a piecewise linear function, so that PMLE can implement any piecewise constant function. Based on Lemma 3.6 (from [24]) one can conclude that the PMLE can compute any function that is computable by a two-hidden-layer net with processors of type H.

From this conclusion and Proposition 2.4, the theorem is proven. □

5. From HH to PMLE and then to multiple inverse controller

5.1. Parametrization via hinging hyperplanes

In this section the relationship between the parameters of the PMLE (Eqs. (2) and (3)) and the HH function approximation (1) are derived, that is, given the number of hinge functions K, the hinges Δ_k, and the hyperplanes β_k^-, β_k^+, the parameters of the PMLE, θ and w, and the structure of the gate functions g_i are derived.

In order to make the description compact and readily programmable with MATLAB, the parameters are written in vector and matrix notation as follows:

$$X = \begin{bmatrix} 1 \\ x(1) \\ \vdots \\ x(M) \end{bmatrix}, \quad D = \begin{bmatrix} \Delta_1(1) & \Delta_2(1) & \cdots & \Delta_K(1) \\ \Delta_1(2) & & & \\ \vdots & \vdots & & \vdots \\ \Delta_1(M+1) & & \vdots & \Delta_K(M+1) \end{bmatrix},$$

$$B_1^+ = \begin{bmatrix} \beta_1^+(1) & \cdots & \beta_K^+(1) \\ \vdots & & \vdots \\ \beta_1^+(M+1) & \cdots & \beta_K^+(M+1) \end{bmatrix}, \quad B^- = \begin{bmatrix} \beta_1^-(1) & \cdots & \beta_K^-(1) \\ \vdots & & \vdots \\ \beta_1^-(M+1) & \cdots & \beta_K^-(M+1) \end{bmatrix}$$

.

The gating function of the PMLE contains a vector Θ_i for each expert that describes its side for each hinge function, and each expert possesses a weight vector W_i as follows:

$$\Theta_i = [\theta_{i1} \quad \cdots \quad \theta_{iK}],$$
$$\theta_{ik} = +1 \text{ if expert } i \text{ belongs to } x^{\mathrm{T}} \cdot \Delta_k \geq 0,$$
$$\theta_{ik} = -1 \text{ if expert } i \text{ belongs to } x^{\mathrm{T}} \cdot \Delta_k < 0,$$
$$W_i = \begin{bmatrix} w_i(1) \\ \vdots \\ w_i(M+1) \end{bmatrix}.$$

For a given input x, the gate can decide which expert describes the function at that point.

$$g_i(x) = \delta_K(\Theta_i \operatorname{sign}(D^{\mathrm{T}}X)) \quad \text{where } \delta_K(u) = \begin{cases} 1, & u = K, \\ 0, & \text{otherwise.} \end{cases}$$

The weights of each expert will be

$$W_i = B^+ H(D^T X) + B^- H(-D^T X) = B^+ H(\Theta_i^T) + B^- H(-\Theta_i^T).$$

The hinging-hyperplanes (HH) parameters are found according to the algorithms in Section 3. The following algorithm transforms the parameters of the HH function approximation to the parameters of the PMLE. This algorithm is an iterative algorithm. There are K hinges and the algorithm inspects them one after the other. For the first hinge two experts are constructed, one for each side of the hinge. The next hinge can be parallel to the first hinge, in which case only one expert will have to split, or the next hinge can intersect with the first hinge and then both experts will have to split. In general for each new hinge, the algorithm checks the position of each expert according to the hinge, and then decides whether it should be split and how to change its parameters. The algorithm uses linear programming (LP) to find the position of each hinge in relation to each expert. The term $x^T \cdot \Delta_k$ where Δ_k is the kth column of the matrix Δ, is positive on one side of the hinge and negative on the other side (see Fig. 3 and Eq. (1)). The maximum and the minimum of this value for the expert are calculated. If both the maximum and the minimum of this term are positive, then the hinge is on one side of the expert, if they are both negative then it is on the other side. LP is chosen for its efficient algorithmic implementation.

Let us phrase the LP problem: In order to find the minimum or maximum of the term $x^T \cdot \Delta$, since the first element in x is 1, which is a constant, we define $\tilde{x}$ as the vector x without the first element, and $\tilde{\Delta}$ as the matrix Δ without the first raw. Our target function is now $\tilde{x}^T \cdot \tilde{\Delta}_k$ and we wish to constrain x to being within the domain of the expert i. That is: for each hinge that has $\theta_{ik} = +1$, $\tilde{x}$ will be such that $x^T \cdot \Delta_k \geq 0$, and for each hinge that has $\theta_{ik} = -1$, $\tilde{x}$ will be such that $x^T \cdot \Delta_k < 0$. We can combine the above to one inequality by multiplying the elements of Δ by the elements of θ. We also have to remember that we should write the constraints on $\tilde{x}$, and finally, we can write the two LP problems

$$\tilde{x}_{\min} = \arg\min_{\tilde{x}}(\tilde{x}^T \cdot \tilde{\Delta}_k) \quad \text{s.t. } A\tilde{x} \leq B \qquad \text{and} \qquad \tilde{x}_{\max} = \arg\max_{\tilde{x}}(\tilde{x}^T \cdot \tilde{\Delta}_k) \quad \text{s.t. } A\tilde{x} \leq B$$

where $x = [1, \tilde{x}^T]^T$, $\Delta = [\Delta_k(1), \tilde{\Delta}_k^T]^T$

$$A = -\begin{bmatrix} \Delta_1(2)\theta_{i1} & \Delta_1(3)\theta_{i1} & \cdots & \Delta_1(M+1)\theta_{i1} \\ \Delta_2(2)\theta_{i2} & & & \\ \vdots & \vdots & & \vdots \\ \Delta_K(2)\theta_{ik} & & \cdots & \Delta_K(M+1)\theta_{ik} \end{bmatrix}, \quad B = \begin{bmatrix} \Delta_1(1)\theta_{i1} \\ \Delta_2(1)\theta_{i2} \\ \vdots \\ \Delta_K(1)\theta_{ik} \end{bmatrix}.$$

Since our interest is in the value of the expression $(x^T \cdot \Delta_k)$ we must calculate

$$Mn = [1, \tilde{x}_{\min}^T] \cdot \Delta_k, \quad Mx = [1, \tilde{x}_{\max}^T] \cdot \Delta_k.$$

In LP the solution might be at ∞ and this can inflict numerical problems. In practice, since we are only interested in the sign of the solution, we add the following constraint for lower and upper bound, $\mathrm{VLB} < x < \mathrm{VUB}$.

Now, we can write the entire algorithm.

Initialization:
For the first hinge, $D = \Delta$, $B^+ = \beta^+$, $B^- = \beta^-$.
Construct two experts as follows:

$$\Theta_1 = [+1], \Theta_2 = [-1]; W_1 = [\beta^+], W_2 = [\beta^-]$$

For each new hinge, k:
For each expert, i:
Find Mn and Mx by solving the LP problem described above
If $Mn > 0$ and $Mx > 0$ the hinge is in one side,

$$W_i = W_i + \beta^+, \Theta_{i,k} = +1,$$

Else If $Mn < 0$ and $Mx < 0$ the hinge is in the other side,

$$W_i = W_i + \beta^-, \Theta_{i,k} = -1$$

Else: the hinge goes through this expert, split to get two experts:

$$W_{N+1} = W_i + \beta^+, \Theta_{N+1,k} = +1,$$

$$W_i = W_i + \beta^-, \Theta_{i,k} = -1$$

End (for expert i)
Add the columns Δ_k, β^+ and β^- to the matrixes D, B^+, B^- respectively.
End (for hinge k)

5.2. Constructing the complete inverse approximation

Now, Once we have the PMLE parameters, corresponding to a piecewise linear approximation of the system, we can use it in order to construct the complete inverse approximation. For the one-dimensional problem, one can invert each expert and get a candidate-solution, which should be validated for being in the expert's range of operation. This idea is illustrated in Fig. 5.

The first two stages in Fig. 5 construct the inverse PMLE (MI-PMLE). The regularization problem is now reduced to a problem of choosing one of the possible solutions. We can give each solution an identification number, call it p, and add this parameter as a regulating input, as described by the block diagram in Fig. 2.

One should notice that even a linear expert could contain a many-to-one mapping in the case of a constant mapping or in the case where there are more inputs than outputs. However there is a simple method using basic linear algebra tools to represent the hyperplane of all the solutions with some real valued parameters (see

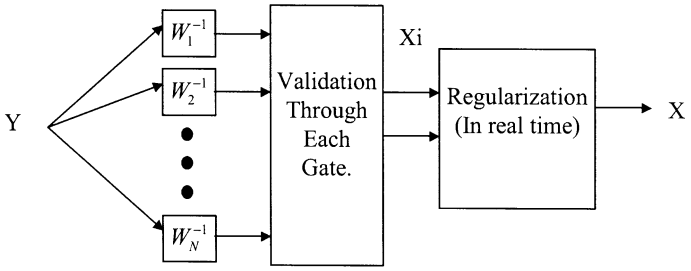


Fig. 5. The process of producing the inverse solutions from the PLME. First each expert produces its own inverse-candidate, then each candidate is checked by its expert gate. The validated inverses are the outputs of the MI-PMLE. Finally, real-time regularization should be executed to choose one of the possible solutions.

Key Theorem 8.26 in [21]). In such cases the regulation vector p will have to be composed of natural numbers in order to choose one of the experts and real number(s) in order to choose a single solution in each linear expert. With this addition, the MI-PMLE can serve as multiple inverse controller to any system that can be approximated by a piecewise linear function with polyhedral boundaries. Further details about the PMLE and the MI-PMLE are available in [15].

Further investigation is needed in order to describe the values of the parameter p (especially in higher-dimensional problems) and in choosing the appropriate solution, but this stage depends on the specific control problem, its constraints and goals. For example, in the case of many motor units around a single joint and in a single muscle, the solution can be chosen as a function of time, i.e. cyclic switching between solutions in order to minimize the fatigue. Another example can be the inverse kinematics of a robotic manipulator, where the solution can be chosen as a function of obstacles in the environment of the manipulator.

6. Simulations

In this section we will illustrate the algorithm by simulation for two examples. The first example is of a smooth function that is not injective, to demonstrate the construction of the MI-PMLE. The second example is of a smooth function in two dimensions.

6.1. *Example 1*: *Smooth function approximation*

This example demonstrates the construction of the complete inverse of a smooth function, which is not an injective. We have drawn 400 examples from the function $y = \sin(x^2)$ where x_i was uniformly distributed in the range $[-2, 2]$. The results of the HH algorithm are given in Figs. 6 and 7.

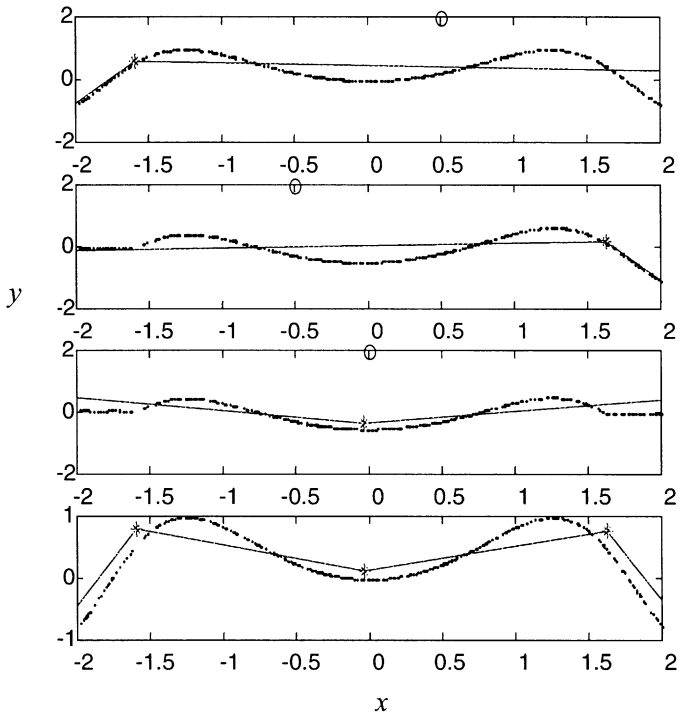


Fig. 6. Simulation of the HFA, fitting three hinges to examples from the function $y = \sin(x^2)$. The dots are the data and the lines are the hinge functions. The circles represent the initial position of the hinge and the stars, the final position of the hinge. The upper plate is the first hinge fitted to the given examples, and then each additional hinge is fitted to the residual error between the samples and the sum of the previous hinges. The lower plate describes the given data and the sum of the three hinges fitted above.

Let us demonstrate the results of the inverse PMLE function, which implements the architecture in Fig. 5. The complete inverse of the target value 0.5 is the following matrix containing four possible solutions:

$$\gg [X] = \text{ipmle}(D, W, \text{teta}, 0.5) \Rightarrow X = [-1.61 \quad -0.73 \quad 1.62 \quad 0.75]$$

and the complete inverse of the target value -0.5 is the following two possible solutions:

$$\gg [X] = \text{ipmle}(D, W, \text{teta}, -0.5) \Rightarrow X = [-1.94 \quad 1.95].$$

Finally, let us use this example further in order to comment on the effect of noise. The HH algorithm is based on a least-squares regression, the basic operation is to fit a hyperplane to the data, therefore as long as the noise has zero mean and there is enough data, the same hyperplane will be fitted (see Fig. 8). The main problem that

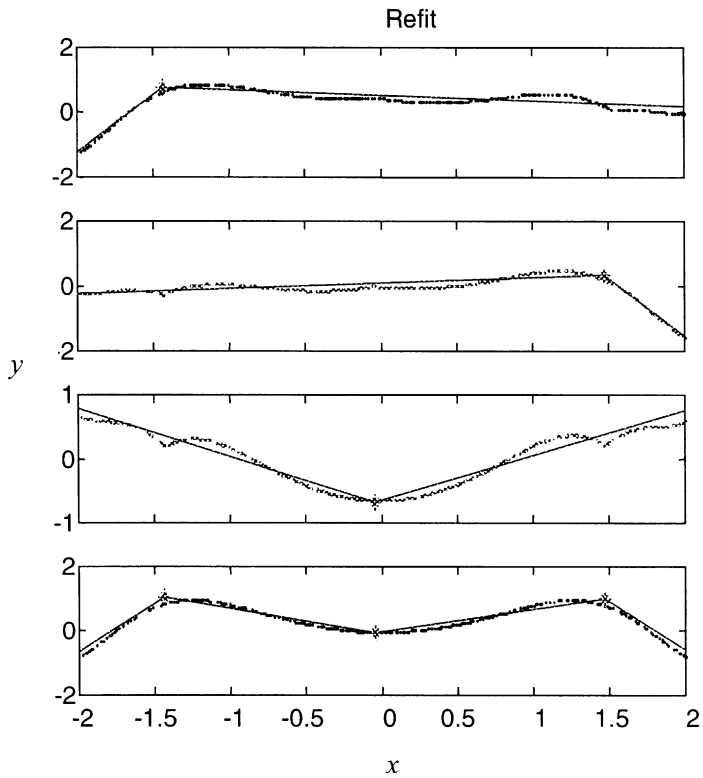


Fig. 7. Simulation of the HFA. Re-fitting the three hinges from Fig. 6 to the examples from the function $y = \sin(x^2)$. This result is after two passes over the refit algorithm. In upper three plates, each hinge is fitted to the residual error between the examples and the sum of the other hinges. The lower plate describes the sum of the three hinge functions over the given data.

can be induced by noise is too many hinges and too many experts. If one select a target error, which is smaller than the noise, then overfitting will occur and new hinges will be generated in order to fit the noise. The solution to this problem is cross validation that is saving some data and stop refitting and adding hinges when the generalization error (the error on the saved data) cease to decrease.

6.2. *Example 2*: *A two-dimensional function approximation*

In this example, we examine a two dimensional (2-D) function. We have chosen to check the algorithm on the following function, which can be presented as a typical control problem. We have drawn 4000 examples from the following function: $y = 4x_2/(1 + x_2^2) + x_1^2 - 2$, where x_1 and x_2 were uniformly distributed in the range $[-2, 2]$. The results of the HH algorithm are presented in Figs. 9–11 .

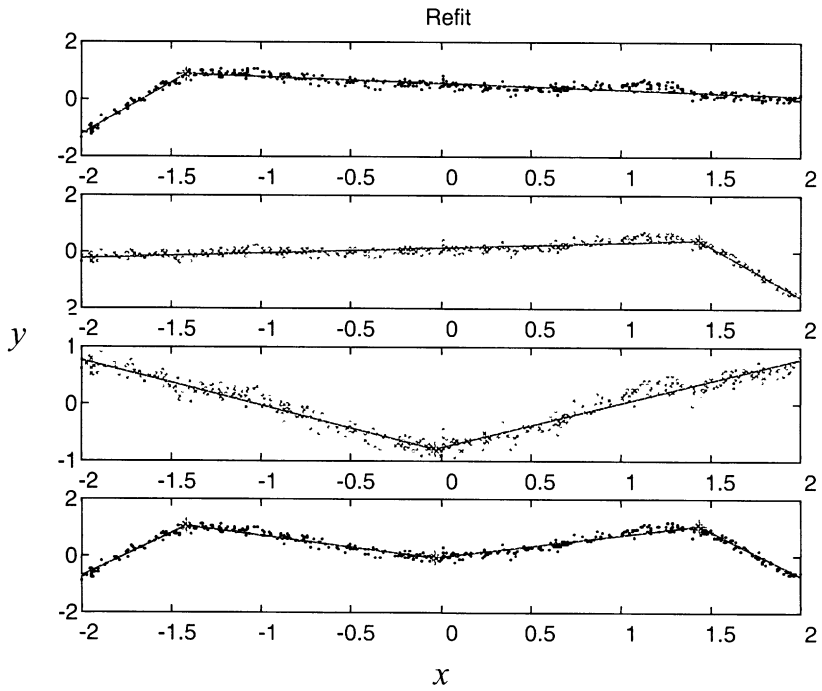


Fig. 8. Simulation of the HFA with added white Gaussian noise. Fitting three hinges to examples from the function $y = \sin(x^2) + N(0, 0.1)$. This result is after two passes over the refit algorithm.

This function was taken from Lee and Lee [19] who considered it as describing a nonlinear system. They tried to identify the dynamics of the difference equation, $y(k+1) = 4y(k)/(1 + y^2(k)) + u(k) - 2$, with their multi-resolution radial basis competitive and cooperative networks. Their method is based on separating the data into hyper-ellipsoidal clusters. The advantage of the PMLE approach presented here is in its ability to identify sharp boundaries between regions of the function and the straightforward parallel implementation as a mixture of experts which enables very fast real-time calculation.

7. Discussion — The virtue of redundancy

The problem of redundancy in the biological system has been known for many years and a large volume of literature has been dedicated to finding the optimization criterion to choose the best single solution (e.g. [5]). In many other problems, the formulation of the question is half the way to the answer. We believe that redundancy should be regarded as a *virtue* rather than a *problem* and therefore the biological system has to find an optimal way to exploit this virtue rather than to "solve this

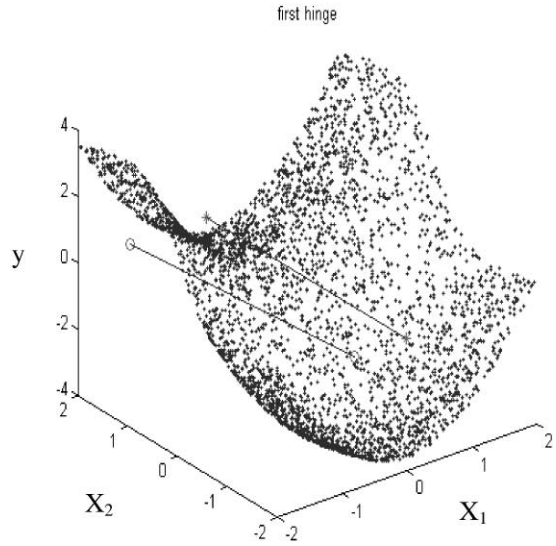


Fig. 9. The examples from the 2-D function and the first hinge's initial position (circles and line) and final position (stars and line).

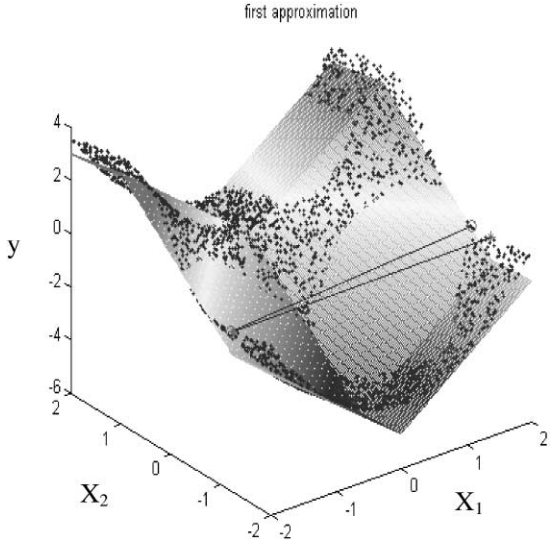


Fig. 10. The hinge function over the examples of the 2-D function.

problem". Therefore, instead of looking for a single optimization criterion that yields a single solution, we suggest a multiple-solutions–multiple-criterion system that can choose different solution in different circumstances. There are two competing views in the literature of motor control: one is the "dynamical" view that assert that the

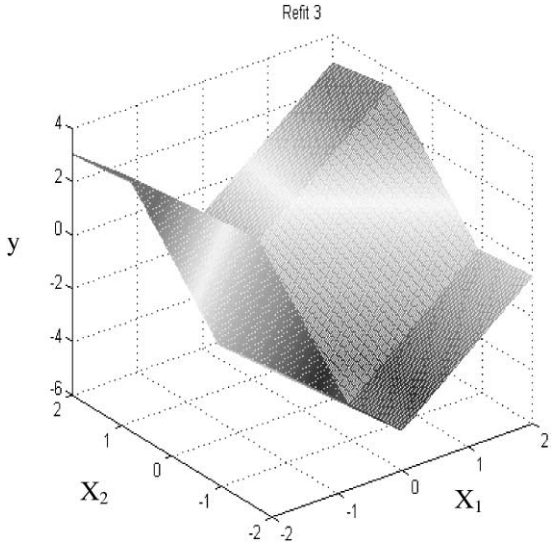


Fig. 11. The final hinge function after refit.

dynamics of the systems "finds" the solution and therefore reduces the redundancy and simplifies the control scheme, see for example the equilibrium point hypothesis (EPH) [2]. The other view is sometimes called "hierarchical", which suggests that the CNS is aware of the details of the controlled system and calculates proper control signals, for example with an inverse model of the controlled system (see [7]). We do not wish to choose one of these views because we think that they both exists at different levels of the control (as suggested by Latash and Anson [17] in their response to commentary). The ultimate solution will probably contain a natural reduction of redundancy due to the dynamical properties of the system (see [13] for an example of the role of the muscles' nonlinear dynamics) and a selection from many possible solutions in real time at a higher level as suggested in this paper.

The multiple inverse PMLE architecture suggests a simple systematic method for the registration of all the solutions. First a piecewise linear estimation is executed, then it is transformed to the PMLE architecture parameters and then each expert is inverted.

In this paper the first stage of finding a piecewise linear estimator is learned by the hinging hyperplanes algorithm of Breiman [3]. Nevertheless, other methods have been suggested recently for the same purpose (see [6,22]). These methods can easily replace the hinging hyperplanes algorithm with minor changes to the transformation algorithm of Section 5 (or no change at all for the algorithms in [22]). The final result of a multiple inverse controller will be exactly the same, and whatever the learning algorithm is, the multiple controller can be straightforwardly implemented in a parallel architecture. A parallel implementation will allow real time calculation of the control signal, and even a real-time switching between criteria for choosing the preferred solution.

With the current decline in the price of hardware, and the availability of parallel computing systems, one should consider multiple controllers that can choose different solutions in different circumstances rather than a single optimization criterion and a single solution. Once we have such multiple controllers, it may also be advantageous to artificially add redundancy to the controlled system in order to enhance its flexibility and reliability as is witnessed in many biological systems. We believe that the Multiple Inverse PMLE controller is a first step in this direction.

8. Conclusions

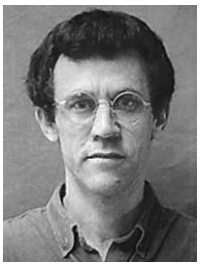
A new architecture for learning the inverse of a redundant system was proposed. This architecture is the polyhedral mixture of linear experts (PMLE), which can learn from examples a piecewise linear approximation of the system and then be easily inverted. The structure of the architecture was presented, its ability to approximate any inverse function was proven, and an algorithm to learn its parameters from examples was described. Finally, the PMLE learning algorithm was demonstrated and the virtue of redundancy was discussed.

References

[1] N. Bernstein, The Coordination and Regulation of Movements, Pergamon Press, Oxford, 1967.
[2] E. Bizzi, N. Hogan, F.A. Mussa-Ivaldi, S. Giszter, Does the nervous system use equilibrium-point control to guide single and multiple joint movements?, Behav. Brain Sci. 15 (1992) 603–613.
[3] L. Breiman, Hinging hyperplanes for regression, classification, and function approximation, IEEE Trans. Inform. Theory 39 (1993) 999–1013.
[4] D.E. DeMers, Learning to invert many-to-one mappings, Ph.D. Thesis, University of California, San Diego, 1993.
[5] T. Flash, N. Hogan, The coordination of arm movements an experimentally confirmed mathematical model, J. Neurosci. 5 (1985) 1688–1703.
[6] D.R. Hush, B. Horne, Efficient algorithms for function approximation with piecewise linear sigmoidal networks, IEEE Trans. Neural Networks 9 (1998) 1129–1141.
[7] G.F. Inbar, A. Yafe, Parameter and signal adaptation in the stretch reflex loop, in: S. Homma (Ed.), Progress in Brain Research, Vol. 44, Elsevier, Amsterdam, 1976, pp. 317–337.
[8] R.A. Jacobs, M.I. Jordan, Learning piecewise control strategies in a modular neural network architecture, IEEE Trans. Systems Man Cybernet. 23 (1993) 337–345.
[9] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixture of local experts, Neural Comput. 3 (1991) 79–87.
[10] M.I. Jordan, Computational aspects of motor control and motor learning. in: H. Heuer, S.W. Keele (Eds.), Handbook of Perception and Action, Vol. 2, Motor Skills. Academic Press, London, 1996.
[11] M.I. Jordan, D.E. Rumelhart, Forward models: supervised learning with distal teacher, Cognitive Sci. 16 (1992) 307–354.
[12] A. Karniel, R. Meir, G.F. Inbar, Polyhedral mixture of linear experts for many-to-one mapping inversion, in: M. Verleysen (Ed.), ESANN'98 Proceedings, D-Facto Publications, Brussels, 1998, pp. 155–160.
[13] A. Karniel, G.F. Inbar, A model for learning human reaching-movements, Biol. Cybernet. 77 (1997) 173–183.

[14] A. Karniel, G.F. Inbar, Human motor control: learning to control a time-varying non-linear many-to-one system, IEEE Trans Systems Man Cybernet. Part C: Appl. Rev. 30 (2000) 1–11.
[15] A. Karniel, Learning motor control of redundant systems, Ph.D. Thesis, Department of Electrical Engineering, Technion–Israel Institute of Technology, 1999.
[16] M. Kawato, H. Gomi, M. Katayama, Y. Koike, Supervised learning for coordinative motor control, Proceedings of the Third NEC Research Symposium, 1993, pp. 126-161.
[17] M.L. Latash, J.G. Anson, What are "normal movements" in a typical populations? Behav. Brain Sci 19 (1996) 55–106.
[18] M.L. Latash, M.T. Turvey (Eds.), Dexterity and its Development, Erlbaum, Aillsdale, LJ, 1996.
[19] S. Lee, J.M. Lee, Nonlinear system control based on multi-resolution radial-basis competitive and cooperative networks, Neurocomputing 9 (1995) 187–206.
[20] B-L. Lu, K. Ito, Regularization of inverse kinematics for redundant manipulators using neural network inversions, IEEE International Conference on Neural Networks Proceedings, 1995, pp. 2726-2731
[21] B. Noble, J.W. Daniel, Applied Linear Algebra, 3rd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[22] P. Pucar, J. Sjöberg, On the hinge-finding algorithm for hinging hyperplanes, IEEE Trans. Inform. Theory 44 (1998) 1310–1319.
[23] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.Y. Glorennec, H. Hjalmarsson, A. Juditsky, Nonlinear black-box modeling in system identification: a unified overview, Automatica 31 (1995) 1691–1724.
[24] E.D. Sontag, Feedback stabilization using two-hidden-layer nets, IEEE Trans. Neural Networks 3 (1992) 981–990.

Amir Karniel was born in 1967 in Jerusalem, Israel. He received the B.Sc. degree (Cum Laude) in 1993 the M.Sc. degree in 1996, and the Ph.D. degree in 2000, all in Electrical Engineering from the Technion-Israel Institute of Technology, Haifa, Israel. He served four years in the Israeli Navy as an electronics technician and worked during his undergraduate studies at Intel Corporation, Haifa, Israel. From 1993 to 1999 he had been a teaching assistant (projects supervisor, tutor and lecturer) in the faculty of Electrical Engineering at the Technion. Since February 2000, he has been a post doctoral fellow at the department of physiology, Northwestern University Medical School, Chicago, Illinois. Dr. Karniel received prizes as distinguished instructor, the E. I. Jury award for excellent students in the area of systems theory, and the Wolf scholarship award for excellent research students. His current research interests are Brain Theory, Neural Networks, Human Motor Control and Motor Learning.



Ron Meir received the B.Sc. degree in Physics and Mathematics from the Hebrew University in Jerusalem in 1982, and the M.Sc. and Ph.D. degrees from the Weizmann Institute in 1984 and 1988, respectively. After two years as a Weizmann research fellow at Caltech he spent a year and a half working at Bellcore on various aspects of neural network design and analysis. He joined the Department of Electrical Engineering at the Technion in 1992. His main research areas include the statistical theory of learning, pattern recognition, time series modeling and neural network design and analysis.

Gideon F. Inbar received the B.Sc. degree from the Technion-Israel Institute of Technology, Haifa, Israel, in 1959, the M.Sc. degree from Yale University, New Haven, CT, in 1963, and the Ph.D. degree from the University of California, Davis, in 1969, all in electrical engineering. In 1970 he joined the Faculty of the Department of Electrical Engineering at the Technion, where he is now Professor and holds the Otto Barth Chair in Biomedical Sciences. From January 1986 he served as Dean of the department of Electrical Engineering for four years. He spent an extended sabbatical at the Harvard Division of Applied Science and School of Public Health, 1977–1978, and shorter periods at Göttingen University, West Germany, at the Centro de Investigacion Del IPN, in Mexico, at the University der BW in Munich and 1991–92 at the Beckman Institute, University of Illinois at Urbana. His major interests are in the areas of biocybernetics and biomedical signal analysis with an emphasis on the neuromuscular system. Dr. Inbar is a member of the Israel Association for Automatic Control, the Israeli Society for Physiology and Pharmacology, and the Israeli Society of Biomedical and Medical Engineering. He is a member and Fellow of the IEEE.