TR－H－045

Multi-Valued Standard Regularization Theory (2): Regularization Networks and Learning Algorithms for Approximating Multi-Valued Functions

Masahiko SHIZAWA

1993. 12. 22

ATR人間情報通信研究所

〒619-02 京都府相楽郡精華町光台2-2　☎07749-5-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

Multi-Valued Standard Regularization Theory (2): Regularization Networks and Learning Algorithms for Approximating Multi-Valued Functions [1]

Masahiko Shizawa
ATR Human Information Processing Research Laboratories

[1]This paper is submitted to IEEE International Conference on Neural Networks (ICNN'94)

Abstract

The Regularization Network (RN) is extended to approximate *multi-valued functions* so that the one-to-h mapping, where h denotes the multiplicity of the mapping, can be represented and learned from a finite number of input-output samples *without clustering operations* on the sample data set. Multi-valued function approximations are useful for learning ambiguous input-output relations from examples. This extension, which we call the Multi-Valued Regularization Network (MVRN), is derived from the Multi-Valued Standard Regularization Theory (MVSRT), which is an extension of the standard regularization theory to multi-valued functions. MVSRT is based on a direct algebraic representation of multi-valued functions. By simple transformation of the unknown functions, we can obtain *linear* Euler-Lagrange equations. Therefore, the learning algorithm for MVRN is reduced to solving a linear system. It is rather surprising that the dimension of the linear system is invariant to the multiplicity h. The proposed theory can be specialized and extended to Radial Basis Function (RBF), Generalized RBF (GRBF), and HyperBF networks of multi-valued functions.

Keywords

Multi-Valued Function, One-to-Many Mapping, Computational Learning, Ambiguous Relations, Function Approximation, Regularization Network, Radial Basis Function Network, GRBF Network, Spline Approximation Network, Feedforward Neural Network.

Acknowledgments

The author thanks Prof. Tomaso Poggio and Dr. Federico Girosi of MIT Artificial Intelligence Laboratories for discussions and helpful comments on the early idea of the work during their visit to ATR. He also thanks Drs. Masa'aki Sato and Hiroshi Ando, and Mr. Hiroyuki Mizutani for their comments on the early Japanese draft version of this paper. Further, the author thanks Mr. Toru Yoshikawa of CSK Corporation for algorithm implementations on CM-5. Finally, he thanks Dr. Yoh'ichi Tohkura, president of ATR Human Information Processing Research Laboratories and Dr. Shigeru Akamatsu, head of department #2 for their support.

Contents

1 Introduction

Learning input-output mappings from examples through feedforward (neural) networks is a fundamental feature of intelligent systems. Many neural network architectures and learning algorithms have been proposed such as multi-layer perceptrons, backpropagations, RBF (Radial Basis Function) networks and regularization networks. From the perspective of realizing complex systems that perform complex tasks such as vision, however, their facilities are still unsatisfactory. The visual task is generally very complex and possibly *ambiguous*. For example, we may encounter a situation of *multiple interpretations* such as *Necker's cube*, and of *perceptual transparency* (**Fig. 1**) . Toward this end, the visual modules for perceiving figures have to be one-to-h mappings, where h denotes the multiplicity of the mappings. In general, for the purpose of learning inverse models from examples, we usually encounter similar problems. Mathematically, this is because the inverse mapping of a single-valued forward mapping is generally a multi-valued mapping (See **Fig. 2** for explanation).

In this paper, we address the problem of learning *multi-valued functions* from examples by regularization networks [9] [10] which are based on the regularization theory [8] [15]. The multi-valued function is defined as a function that produces possibly multiple different values at each point in the input space. The multi-valued function is unlike the vector-valued function[6], since it does not care about the order of multiple values in the outputs. Further, the examples for learning are given with single-valued outputs. If we use conventional neural networks for this learning task, we need to cluster the learning sample data into groups that correspond to different hypersurfaces. This process is computationally burdensome and difficult in general especially with noisy learning samples. Conventional feedforward neural networks have been concentrating on building single-valued mappings from input to output. Instead, to learn multi-valued functions, we might try to build multiple networks separately from manually separated learning sample data and then combine them, or competitively learn the multiple input-output mappings through a mixture of multiple 'expert' networks [3]. The latter approach requires optimization techniques such as Simulated Annealing [5] and the EM (Expectation and Maximization) algorithm [1] for non-convex general functions.

This paper provides a quantum leap from the conventional approaches.

We represent a multi-valued function by a single direct algebraic equation. Therefore, the learning problem becomes a reconstruction of a single hypersurface. We formulate a standard regularization problem for the algebraic representation of the multi-valued functions. Using this formalism, the Euler-Lagrange equation of the energy minimization problem becomes linear with respect to the unknown functions. This means that the techniques for deriving the regularization networks [9] [10] can be applied directly to multi-valued function approximation.

The regularization networks have been proven to be capable of approximating arbitrarily well any continuous function on a compact subset of R^n [2]. Therefore, they may be considered as alternatives to feedforward multilayer perceptrons. It should be noted, however, that our contribution is not restricted to regularization networks, since the basic idea can be applied to other types of feedforward neural networks that approximate input-output mappings from examples, and can extend such networks enabling them to learn multi-valued mappings.

The paper is organized as follows. First, in section 2, the standard regularization theory, which is the leading principle of the regularization networks, is extended to approximate multi-valued functions. The extension is based on a direct representation of the multi-valued function using tensor product (Kronecker's product). We call this extension the Multi-Valued Standard Regularization Theory (MVSRT). Section 3 derives the regularization networks and their learning algorithm for approximating multi-valued functions (MVRN) based on MVSRT. Also, an approximation to MVRN, called MV-GRN and having a smaller number of hidden units than learning samples, is proposed. Section 4 verifies MVRN by a simulation using Gaussian RBF. Section 5 discusses important remaining issues for future investigations and concludes this paper.

2 Multi-Valued Standard Regularization Theory (MVSRT)

2.1 Standard Regularization Theory and Regularization Networks

The Standard Regularization Theory is a general framework for solving ill-posed inverse problems in various scientific and engineering problems [8] [15]. The learning of input-output mappings from examples can be formulated by using the regularization principle. The energy functional for approximating a scalar mapping $f : R^n \mapsto R$ is written as follows.

$$E^{(1)}[f] = \sum_{i=1}^{N} \left(y_{(i)} - f(\boldsymbol{x}_{(i)})\right)^2 + \lambda \|Sf\|^2, \tag{1}$$

where S is a regularization (smoothness) operator, λ is a regularization parameter, and $\|\quad\|$ is a norm of a functional space. $(\mathbf{x}_{(i)}, y_{(i)}) \in R^n \times R$ denotes the i-th data where $i = 1, 2, \cdots, N$. The second term, called the regularization term, determines the smoothness of the mapping which is interpretated as *generalization* in the learning theory. By applying a variational method to this formalism, Poggio & Girosi [9] [10] derived the Regularization Network (RN) which is a three-layer network for approximating smooth input-output functions from examples. Several learning methods including Radial Basis Function networks and spline approximations can be derived as special cases of the regularization network. In the subsequent sections, we extend this formalism into a framework for approximating *multi-valued functions.*

2.2 Direct Representation of Multi-Valued Functions

The basis of our approach is quite fundamental; the direct extension of the input-output relation $\boldsymbol{f} : R^n \mapsto R^m$ in the form $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ into a multi-valued function. This was transferred from our previous work on multiple overlapping smooth surface reconstruction from sparse depth data in computational vision [12].

2.2.1 General representation of multi-valued functions

The general form of the basic relation for h-valued mappings from R^n to R^m can be written as [12],

$$(\boldsymbol{y} - \boldsymbol{f}_1(\boldsymbol{x})) \otimes (\boldsymbol{y} - \boldsymbol{f}_2(\boldsymbol{x})) \otimes \cdots \otimes (\boldsymbol{y} - \boldsymbol{f}_h(\boldsymbol{x})) = \boldsymbol{0}, \tag{2}$$

where $\boldsymbol{f}_i(\boldsymbol{x})$ $(i = 1, 2, \cdots, h)$ denotes component single-valued functions, and $\otimes$ denotes tensor product (Kronecker's product). This is a mathematically strict representation of the multi-valued function. We do not need any auxiliary parameter or label of any kind other than original functions $\boldsymbol{f}_i(\boldsymbol{x})$. This says that each point $(\boldsymbol{x}, \boldsymbol{y})$ must be on at least one of the h hypersurfaces $\boldsymbol{y} = \boldsymbol{f}_i(\boldsymbol{x})$ $(i = 1, 2, \cdots, h)$.

There are several points to note in Eq. (2). The number of element equations in Eq.(2) is m^h. This number is much higher than the freedom of the functions (m). One way of reducing this redundancy is to use the fact that any two functions $\boldsymbol{f}_k(\boldsymbol{x})$ and $\boldsymbol{f}_l(\boldsymbol{x})$ $(k \neq l)$ can be exchanged with each other from the original meaning of the problem. Since Kronecker's product is not commutative, Eq. (2) does not represent this fact. We can incorporate this exchangeability of the functions by using *symmetrization* of the tensor product [14]. Then, the number of independent element equations can be computed by the repeated combination ${}_mH_h = {}_{m+h-1}C_h = \frac{(m+h-1)!}{h!(m-1)!}$. It still seems redundant in comparison with the freedom of the values of the functions. However, this remaining redundancy guarantees the uniqueness of the representation.

The tensor product representation Eq. (2) of the multi-valued function is motivated from quantum mechanics of elementary particles. It has a very interesting analogy to the quantum mechanical representation for the system of multiple elementary particles of the same kind (cf. [7] [14]). In the following, however, we concentrate on the scalar function approximations, i.e., the case of $m = 1$, since they play fundamental roles even in general cases. More general cases including multi- and vector-valued functions are described in Refs. [12] [13].

2.2.2 Direct representation of h-valued scalar function

A multi-valued scalar function can be written in much more simple formulae. We define an h-valued function as follows.

$$\Lambda^{(h)}(\boldsymbol{x}, y) = \prod_{i=1}^{h} (y - f_i(\boldsymbol{x}))$$
$$= F_1^{(h)}(\boldsymbol{x}) + yF_2^{(h)}(\boldsymbol{x}) + \cdots + y^{k-1}F_k^{(h)}(\boldsymbol{x}) + \cdots + y^{h-1}F_h^{(h)}(\boldsymbol{x}) + y^h = 0, \quad (3)$$

where,

$$\begin{aligned} F_1^{(h)}(\boldsymbol{x}) &= (-1)^h f_1(\boldsymbol{x}) f_2(\boldsymbol{x}) \cdots f_h(\boldsymbol{x}); \cdots; \\ F_{h-1}^{(h)}(\boldsymbol{x}) &= \sum_{i_1=1}^{h} \sum_{i_2=i_1+1}^{h} f_{i_1}(\boldsymbol{x}) f_{i_2}(\boldsymbol{x}); \\ F_h^{(h)}(\boldsymbol{x}) &= -\sum_{i=1}^{h} f_i(\boldsymbol{x}), \end{aligned} \quad (4)$$

are elementary symmetrical expressions of the functions $f_i(\boldsymbol{x})$. Eq. (3) is an h-degree univariate algebraic equation with respect to y, and is a special case of Eq. (2).

These substitutions of the unknown functions greatly simplify the following discusssions, since Eq. (3) is linear with respect to functions $F_k^{(h)}(\boldsymbol{x})$ $(k = 1, 2, \cdots, h)$.

2.3 Standard Regularization of Multi-Valued Functions

The regularization problem for multi-valued functions can then be formulated by minimization of the following functional with respect to $F_k^{(h)}$ $(k = 1, 2, \cdots, h)$.

$$E^{(h)}[F_1^{(h)}, F_2^{(h)}, \cdots, F_h^{(h)}] = \sum_{i=1}^{N} \{\Lambda^{(h)}(\boldsymbol{x}_{(i)}, y_{(i)})\}^2 + \sum_{k=1}^{h} \lambda_k \| S_k F_k^{(h)}(\boldsymbol{x}) \|^2, \quad (5)$$

where S_k and λ_k are the regularization operator and regularization parameter for $F_k^{(h)}(\boldsymbol{x})$, respectively. It should be emphasized that this is a *standard* regularization problem, i.e., convex quadratic minimization, since we adopt regularization on functions $F_k^{(h)}(\boldsymbol{x})$ instead of original functions $f_i(\boldsymbol{x})$. We call

this extension the Multi-Valued Standard Regularization Theory (MVSRT) [12] [13]. Therefore, the solution to this problem can be easily obtained by standard regularization techniques as will be shown in the next section.

3 Multi-Valued Regularization Networks

MVRN consists of two network modules (**Fig. 3**). The first module maps the input vector $\boldsymbol{x}$ into $F_k^{(h)}(\boldsymbol{x})$ $(k = 1, 2, \cdots, h)$, while the subsequent module maps $F_k^{(h)}(\boldsymbol{x})$ into $f_i(\boldsymbol{x})$. In the following, we derive MVRN and its learning algorithms from MVSRT.

3.1 h-Valued Regularization Networks and Learning Algorithm

For simplicity, we assume $S = S_k$ and $\lambda = \lambda_k$ for $k = 1, 2, \cdots, h$ in the following discussions.

3.1.1 Derivation of the network

The Euler-Lagrange equation for the minimization problem of Eq. (5) is

$$\sum_{i=1}^{N} \left(y_{(i)}\right)^{k-1} \left\{\Lambda(\boldsymbol{x}, y_{(i)})\right\} \delta(\boldsymbol{x} - \boldsymbol{x}_{(i)}) + \lambda \hat{S} S F_k^{(h)}(\boldsymbol{x}) = 0, \tag{6}$$

where $k = 1, 2, \cdots, h$. This equation can be solved by using the Green's function $K(\boldsymbol{x}, \boldsymbol{x}')$ of operator $\hat{S}S$ defined as

$$\hat{S} S K(\boldsymbol{x}, \boldsymbol{x}') = \delta(\boldsymbol{x} - \boldsymbol{x}'), \tag{7}$$

where $\hat{S}$ is the adjoint operator of S. We have the following representation of MVRN.

$$F_k^{(h)}(\boldsymbol{x}) = \sum_{i=1}^{N} r_i^{(h)} \left(y_{(i)}\right)^{k-1} K(\boldsymbol{x}, \boldsymbol{x}_{(i)}), \tag{8}$$

where

$$r_i^{(h)} = -\lambda^{-1} \Lambda(\boldsymbol{x}_{(i)}, y_{(i)}). \tag{9}$$

Equation (8) shows that functions $F_k^{(h)}(\boldsymbol{x})$ can be represented as linear combinations of Green's functions $K(\boldsymbol{x}, \boldsymbol{x}_{(i)})$. It should be noted that *the number*

of weight parameters is equal to the number of examples N, and does not depend on the multiplicity h of the mapping. The weight parameters $r_i^{(h)}$ $(i = 1, 2, \cdots, N)$ are shared by all functions $F_k^{(h)}(\boldsymbol{x})$ $(k = 1, 2, \cdots, h)$.

3.1.2 Learning algorithm for MVRN

We can derive a learning algorithm for determining weight parameters $r_i^{(h)}$ in a similar way to the derivation of the regularization networks in Ref.[9].

Substituting $F_k^{(h)}(\boldsymbol{x}_{(i)})$ in Eq. (9) by Eq. (8) and rearranging terms result in the following N-dimensional linear system with respect to the weights $r_i^{(h)}$.

$$\boldsymbol{K}^{(h)}\boldsymbol{r}^{(h)} + \boldsymbol{z}^{(h)} = 0, \tag{10}$$

where,

$$\begin{aligned}
\boldsymbol{K}^{(h)} &= \left(K_{ij}^{(h)}\right) = \left(\left\{\sum_{k=1}^{h}\left(y_{(i)}y_{(j)}\right)^{k-1}\right\}K(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}) + \lambda\delta_{ij}\right), \\
\boldsymbol{r}^{(h)} &= \left(r_1^{(h)}, r_2^{(h)}, \cdots, r_N^{(h)}\right)^T, \\
\boldsymbol{z}^{(h)} &= \left(\left\{y_{(1)}\right\}^h, \left\{y_{(2)}\right\}^h, \cdots, \left\{y_{(N)}\right\}^h\right)^T,
\end{aligned} \tag{11}$$

where δ_{ij} denotes Kronecker's delta. Since the weight parameters are shared by h functions $F_k^{(h)}(\boldsymbol{x})$, the dimension of the linear system is invariant to the multiplicity h. This is a particularly significant property, since the computational complexity for solving an N-dimensional linear system is approximately equaivalent to N^3.

3.1.3 Multi-Valued Generalized Regularization Networks (MVGRN)

MVRN can produce an exact solution of the standard regularization problem (Eq. (5)). However, the complexity grows rapidly with increasing of number of examples. We can consider the MVRN defined by Eq. (8) to be linear combinations of 'basis functions' $K(\boldsymbol{x}, \boldsymbol{x}_{(i)})$ whose centers are at $\boldsymbol{x}_{(i)}$ $(i = 1, 2, \cdots, N)$. Poggio & Girosi [9] proposed the Generalized Regularization Network (GRN) in which the number of basis functions $K(\boldsymbol{x}, \boldsymbol{x}')$ is less than the number of learning samples N. We denote the (fixed) centers of the

basis functions as $\boldsymbol{t}_j$ $(j = 1, 2, \cdots, M < N)$ where M is the number of basis functions. Then, the Multi-Valued Generalized Regularization Network (MVGRN) $\tilde{F}_k^{(h)}(\boldsymbol{x})$ can be written as

$$\tilde{F}_k^{(h)}(\boldsymbol{x}) = \sum_{j=1}^{M} \tilde{r}_{k,j}^{(h)} K(\boldsymbol{x}, \boldsymbol{t}_{(j)}), \tag{12}$$

where, in contrast to MVRN, we cannot share the weight parameters between functions $\tilde{F}_k^{(h)}(\boldsymbol{x})$ for $k = 1, 2, \cdots, h$. Thus, we denote the hM weight parameters as $\tilde{r}_{k,j}^{(h)}$ $(k = 1, 2, \cdots, h; j = 1, 2, \cdots, M)$. The learning algorithm becomes an hM-dimensional linear system.

$$\tilde{\boldsymbol{K}}^{(h)} \tilde{\boldsymbol{r}}^{(h)} + \tilde{\boldsymbol{z}}^{(h)} = \boldsymbol{0}, \tag{13}$$

where,

$$\begin{aligned}
\boldsymbol{K}^{(h)} &= \begin{bmatrix} \boldsymbol{D}_1^T \boldsymbol{D}_1 + \lambda \boldsymbol{J} & \boldsymbol{D}_1^T \boldsymbol{D}_2 & \cdots & \boldsymbol{D}_1^T \boldsymbol{D}_h \\ \boldsymbol{D}_2^T \boldsymbol{D}_1 & \boldsymbol{D}_2^T \boldsymbol{D}_2 + \lambda \boldsymbol{J} & \cdots & \boldsymbol{D}_2^T \boldsymbol{D}_h \\ \cdots\cdots & \cdots\cdots & \cdots & \cdots\cdots \\ \boldsymbol{D}_h^T \boldsymbol{D}_1 & \boldsymbol{D}_h^T \boldsymbol{D}_2 & \cdots & \boldsymbol{D}_h^T \boldsymbol{D}_h + \lambda \boldsymbol{J} \end{bmatrix}, \\
\tilde{\boldsymbol{r}}^{(h)} &= \left(\tilde{r}_{1,1}, \cdots, \tilde{r}_{1,M}, \tilde{r}_{2,1}, \cdots, \tilde{r}_{2,M}, \cdots, \tilde{r}_{h,1}, \cdots, \tilde{r}_{h,M}\right)^T, \\
\tilde{\boldsymbol{z}}^{(h)} &= \left(\left(\boldsymbol{z}^{(h)}\right)^T \boldsymbol{D}_1, \left(\boldsymbol{z}^{(h)}\right)^T \boldsymbol{D}_2, \cdots, \left(\boldsymbol{z}^{(h)}\right)^T \boldsymbol{D}_h, \right)^T, \\
(\boldsymbol{D}_k)_{ij} &= \left(y_{(i)}\right)^{k-1} K(\boldsymbol{x}_{(i)}, \boldsymbol{t}_{(j)}), \quad (\boldsymbol{J})_{ij} = K(\boldsymbol{t}_{(i)}, \boldsymbol{t}_{(j)}), \\
\boldsymbol{z}^{(h)} &= \left(\left\{y_{(1)}\right\}^h, \left\{y_{(2)}\right\}^h, \cdots, \left\{y_{(N)}\right\}^h\right)^T.
\end{aligned} \tag{14}$$

3.1.4 Decomposition network

The second network for transforming functions $F_k^{(h)}(\boldsymbol{x})$ into $f_i(\boldsymbol{x})$ may be realized in two ways.

Recurrent network One is the recurrent network which numerically solves the h-degree algebraic equation (3) with respect to y. In our case, it is convenient for us if all of the solutions can be obtained simultaneously. The iterative method known as the Durand-Kerner's method [4] meets our purpose. The following iterative update equation solves Eq. (3) simultaneously

for all h solutions.

$$f_j^{[k+1]}(\boldsymbol{x}) = f_j^{[k]}(\boldsymbol{x}) - \Lambda^{(h)}(\boldsymbol{x}, f_j^{[k]}(\boldsymbol{x})) \left\{ \prod_{i=1, i\neq j}^{h} \left(f_j^{[k]}(\boldsymbol{x}) - f_i^{[k]}(\boldsymbol{x}) \right) \right\}^{-1}, \quad (15)$$

where k denotes an iteration counter. This process has been proven to be equivalent to the Newton-Raphson method applied to the simultaneous algebraic equation, Eq. (4) [4]. This network may become unstable for a multiple-root case.

Feedforward network The second approach is the use of another regularization network, or any feedforward neural networks for approximately realizing the inverse mapping. This network can be build independent of the first network module. We can embed a function for detecting multiple roots in the decomposition network.

4 Simulations using Gaussian Radial Basis Function

We have implemented a Gaussian RBF version of MVRN and have tested several noisy data configurations. For Gaussian RBF, the regularization operator is defined so that Green's function $K(\boldsymbol{x}, \boldsymbol{x}')$ becomes an n-dimensional isotropic Gaussian of $(\boldsymbol{x} - \boldsymbol{x}')$: $K(\boldsymbol{x}, \boldsymbol{x}') = A \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2 / 2\sigma^2\right)$ where A is a constant. In this case, operator S is a pseudo differential operator [9]. The decomposition network was implemented by a recurrent network incorporating Durand-Kerner's method. **Figure 4** shows one of the results for a three-valued MVRN (Gaussian RBF version). In this simulation, the parameters were set as $\sigma = 0.1$, $\lambda = 10.0$. The dimension of the input and output space were both one-dimensional, and the multiplicity of MVRN was $h = 3$. The number of sample data was 500 for each surface.

5 Conclusions

A fundamental concept and techniques have been proposed for extending conventional regularization networks, including Radial Basis Function networks as a special case, to h-valued function approximation. The learning

algorithm turns out to be a linear system with no clustering operations on the learning sample data set. This property is quite significant in regard of computational complexity and robustness of learning. The complexity of the learning is almost the same as conventional single-valued function approximations.

The idea behind our approach is quite different from 'competitive' learning paradigms. The underlying idea can be dubbed as 'coexistence' rather than competition. This was strongly motivated from quantum mechanical representations of multiple elementary particle systems. The algebraic representation of the multi-valued function in Eq. (2) represents this idea in rigorous mathematics.

There are several important problems for future investigations:

1. The multiplicity h should be determined automatically in the learning step. This may be possible if we can estimate the *rank* of the linear system for learning.

2. For the purpose of learning inverse models from examples like the mapping depicted in **Fig. 2**, the network needs to handle point-wise changes of multiplicity h. This may be possible if the decomposition network is a feedforward network with a function for determining multiple roots from inputs $F_k^{(h)}(\boldsymbol{x})$. Another possibility is placing a gating network to select appropriate outputs from among MVRNs of different multiplicity h.

3. Regarding MVGRN, we can incorporate moving centers and norms of Green's functions in a similar way with GRBF and HyperBF networks [9].

4. The direct representation of the multi-valued function (2) can be used for h single-valued regularization networks to separately approximate each hypersurface of an h-valued function. For this purpose, we can derive a learning algorithm based on minimization of the residual of Eq. (2). The learning algorithm will not be a simple linear system, but a set of linear systems each of whose coefficient matrices nonlinearly depend on solutions of the others. Further, Eq. (2) is not dedicated only to the regularization networks, but is also available for other feedforward neural networks including multilayer perceptrons.

References

[1] Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J Roy Statis Soc 39(1): 1-38

[2] Girosi F, Poggio T (1990) Networks and the best approximation property. Biol Cybern 63: 169-176

[3] Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural Comput 3: 79-87

[4] Kerner IO (1966) Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen. Numer Math 8: 290-294

[5] Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220: 219-227

[6] Mussa-Ivaldi FA, Gandolfo F (1993) Networks that approximate vector-valued mappings. Proc IEEE Int Conf Neural Networks, San Francisco, CA, pp.1973-1978

[7] Penrose R (1989) The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics. Oxford Univ Press, Oxford, UK

[8] Poggio T, Torre V, Koch C (1985) Computational vision and regularization theory. Nature 317: 314-319

[9] Poggio T, Girosi F (1990) Networks for approximation and learning. Proc IEEE 78(9): 1481-1497

[10] Poggio T, Girosi F (1990) Regularization algorithms that are equivalent to multilayer networks. Science 247: 978-982

[11] Poggio T, Edelman S, Fahle M (1992) Learning of visual modules from examples: A framework for understanding adaptive visual performance. CVGIP: Image Understanding 56(1): 22-30

[12] Shizawa M (Oct. 1993) Multi-valued standard regularization theory (1): Global reconstruction of multiple transparent surfaces via massively parallel relaxation algorithms. ATR Technical Report TR-H-037

[13] Shizawa M (Dec. 1993) Extension of the standard regularization theory into multi-valued functions: Multi-valued regularization networks and learning algorithms (in Japanese). ATR Technical Report TR-H-039

[14] Sudbery A (1986) Quantum mechanics and the particles of nature. Cambridge Univ. Press

[15] Tikhonov AN, Arsenin VY (1977) Solutions of ill-posed problems. W.H.Winston, Washington DC

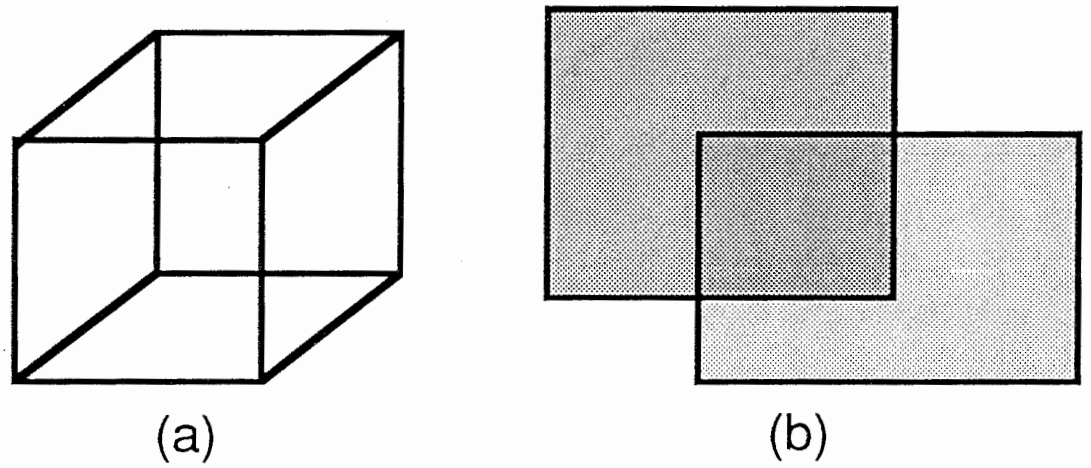

Figure 1: Examples of ambiguous and multiple visual perception.
These are examples where multi-valued mappings are essential components for visual modules. (a) Necker's cube (ambiguous visual perception). (b) Multiple overlapping transparent surfaces (perceptual transparency).

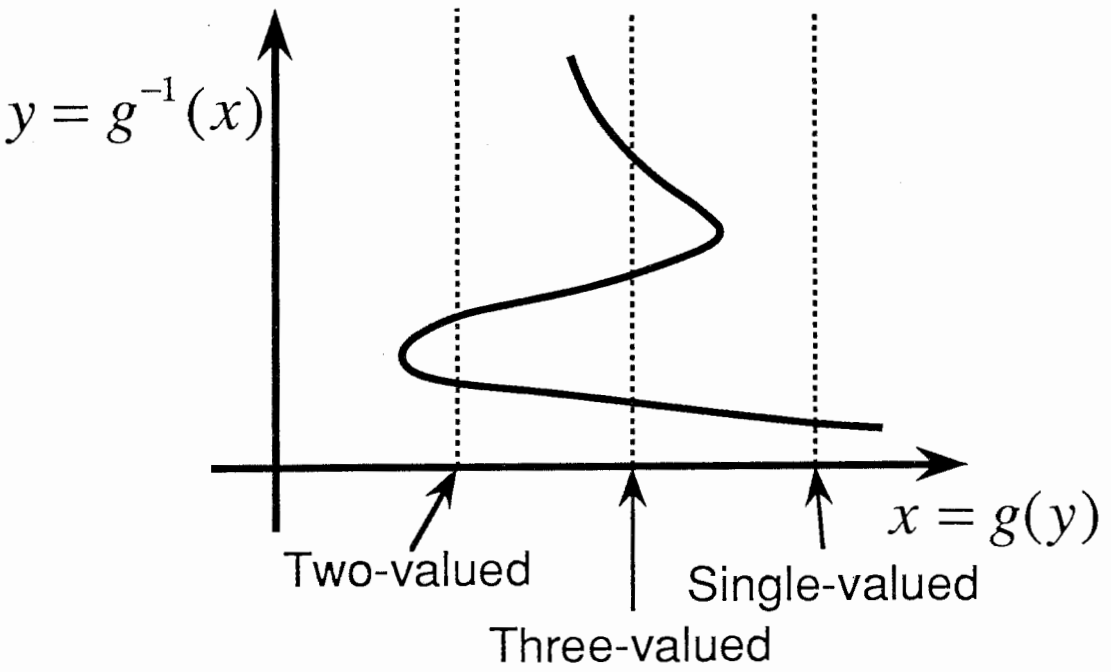

Figure 2: Example illustrating that inverse mapping is generally multi-valued mapping.
Although the forward mapping $g : y \mapsto x$ is single-valued, the inverse mapping $g^{-1} : x \mapsto y$ is multi-valued. This figure illustrates the case where the inverse mapping is a combination of 1-, 2- *and* 3-*valued mappings.*

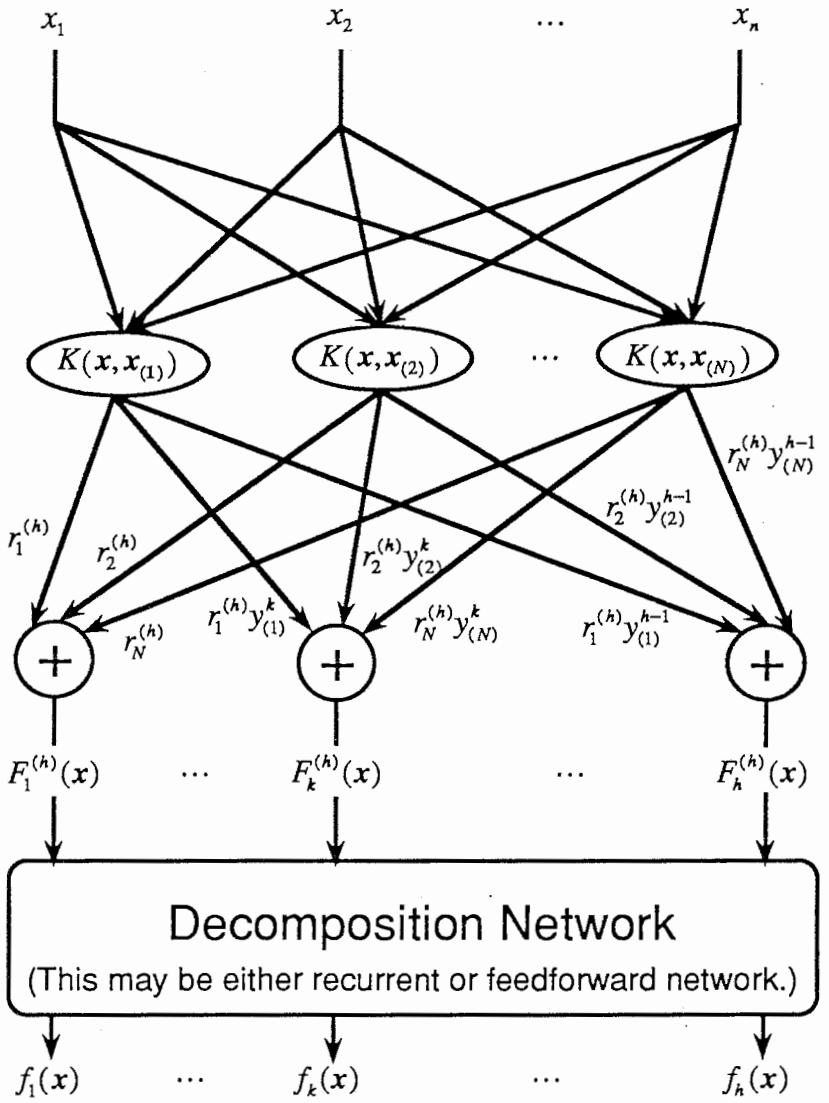

Figure 3: h-valued regularization network.

The multi-valued regularization network (MVRN) has two network modules. The first module maps the input vector $\boldsymbol{x}$ *into* $F_k^{(h)}(\boldsymbol{x})$ $(k = 1, 2, \cdots, h)$ *which is an intermediate representation, while the second module, a decomposition network, maps* $F_k^{(h)}(\boldsymbol{x})$ *into* $f_i(\boldsymbol{x})$*. The second module solves an* h*-degree univariate algebraic equation whose coefficients are* $F_k^{(h)}(\boldsymbol{x})$*. The second module can be either a recurrent network which is equivalent to a numerical solver of algebraic equations, or another regularization network, which approximately realizes this function. The learning of MVRN is to determine weights* $r_i^{(h)}$ *for* $i = 1, 2, \cdots, N$*. Note that the number of unknown weights is invariant to the multiplicity* h*.*

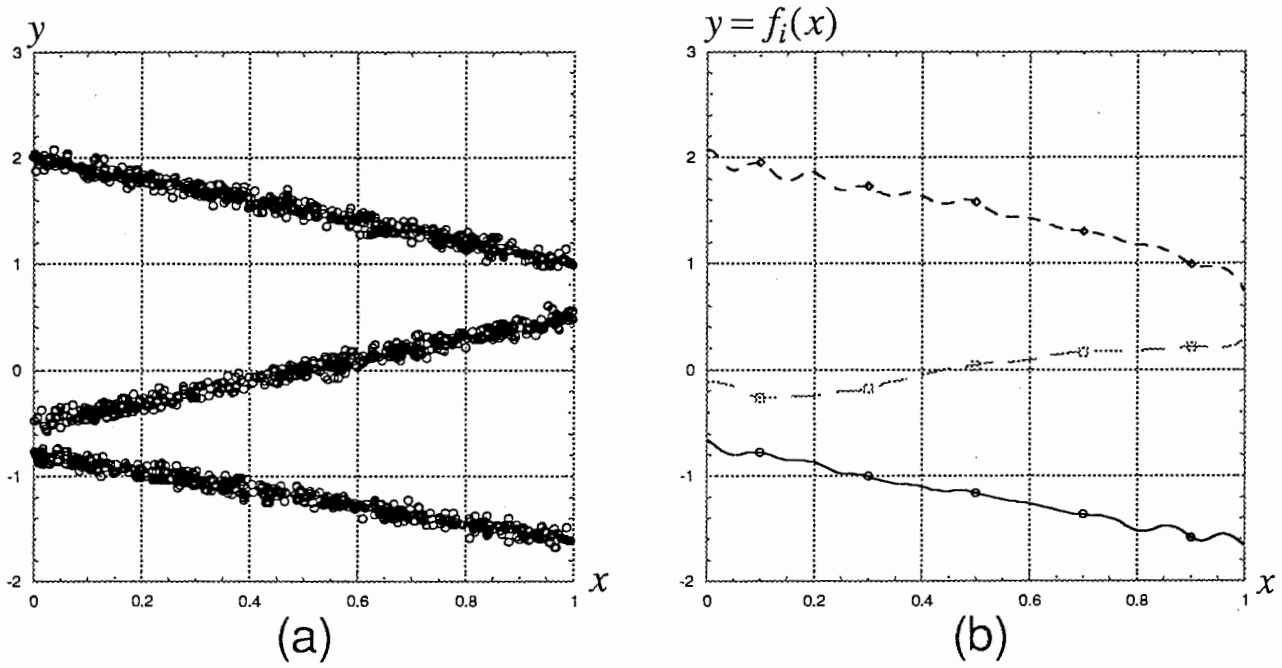

Figure 4: Simulation results for three-valued regularization network. *This is one example for MVRN (Gaussian RBF version) in the case of a three-valued $R \mapsto R$ mapping. (a) Sample data set with additive Gaussian noise (standard deviation = 0.02). (b) Three-valued mapping which was learned from the sample data set. The parameters were set as* $\sigma = 0.1$, $\lambda = 10.0$.