Approximations of Nonlinear Functions by Feed-Forward Neural Networks[1]

Yoshio Takane (McGill University)
Yuriko Oshima-Takane (McGill University)
Thomas R. Shultz (McGill University)

1 Introduction & Summary

Neural network (NN) models are very popular in artificial intelligence, pattern recognition, cognitive psychology, etc. Feed-forward networks may be viewed as approximating nonlinear functions that connect inputs to outputs (e.g., Ripley, 1993). They are known to be robust and efficient approximators of nonlinear functions (e.g., Hornik, Stinchcombe, & White, 1989). We analyze how the approximations are done using a variety of multivariate and graphical techniques (Takane, Oshima-Takane, & Shultz, 1994; Oshima-Takane, Shultz, & Takane, forthcoming). The particular network architecture we are interested in is the cascade correlation (CC) learning network (Fahlman & Lebiere, 1990) which is capable of dynamically growing nets to adapt to more complicated problems. We look at how the learning and representation of knowledge occur in the CC networks as it performs a variety of tasks. We also examine the generalization capability and the effect of environmental bias in the training.

2 Cascade Correlation (CC) Learning Network

Artificial NN networks consist of a set of units, each performing a simple operation. They receive inputs from other units, compute activations based on the inputs, and send out the activations to other units according to the connection strengths. A network of such units interconnected with each other, however, turned out to be quite a powerful computational device. A multilayered network can approximate any function with a finite number of discontinuities, arbitrarily well, given sufficient units in the hidden layers (Hornik et al., 1989).

In the CC learning architecture no *a priori* net topology has to be specified. It starts as a net without hidden units, and it adds hidden units to improve its performance until a satisfactory degree of performance is reached. Hidden units are added one at a time so that all pre-existing units are connected to new ones. Input units are directly connected to output units (cross connections) as well as to all hidden units. The cross connections often simplify the constructed net solutions by capturing linear effects of bias and input units in the simplest possible way. When a new hidden unit is recruited, incoming weights to the new unit are determined by a heuristic method, and are fixed throughout the rest of the learning process. This avoids the necessity of back-propagating error, and leads to faster and more stable convergence.

CC nets were used to implement a wide variety of successful psychological models, particularly in cognitive and semantic development (e.g., Shultz et al., 1994).

3 The Continuous *XOR* Problem

There are a number of bench mark tasks we could use. For illustration, we consider the continuous *xor* problem. This problem is simple enough to deduce what the target function is, and yet complicated enough to warrant network modelling. It requires an interaction between two input variables. Capturing interaction effects between input variables without explicitly being instructed to do so is one of the main advantages of NN models.

The continuous *xor* problem has two input variables, x_1 and x_2, each ranging from 0.1 to 1.0, and one binary output variable, y. The problem is to discriminate two groups of input patterns. When both x_1 and x_2 are greater than or both smaller than 0.55, $y = -0.5$; When only one of the two

[1]In the *Proceedings of the Japan Classification Society Meeting*, Tokyo, December, 1994. pp. 26-33

variables is greater than but the other smaller than 0.55, $y = 0.5$. The optimal discriminating function for this problem is

$$y = f\{-c(x_1 - 0.55)(x_2 - 0.55)\} - 0.5, \quad (1)$$

where f is a sigmoid function, $f(t) = 1/\{1 + exp(-t)\}$, and c goes to infinity. This function is depicted in Figure 1. A CC net can learn to approximate this function, but how? Obviously, this function needs an interaction effect between x_1 and x_2. Figure 2 depicts a net approximation to this function obtained by the CC learning algorithm. The approximation looks good, but how was it achieved? This is the sort of question we address in the present paper.

4 Activations and Contributions

The CC algorithm constructs a net and estimates connection weights based on a sample of training patterns. For the continuous *xor* problem, the sample of training patterns we used are 100 triplets of the form, (x_1, x_2, y), both x_1 and x_2 ranging from 0.1 to 1.0 in steps of 0.1, and $y = -0.5$ or 0.5. In the example depicted in Figure 2, the CC algorithm constructed a net with four hidden units, starting from a net with one bias, two input units and one output unit only.

For each input pattern, a unit in a trained net sends contributions to units it is connected to. A contribution is defined as the product of the activation of the sending unit and the connection weight. The receiving unit forms an activation by summing contributions from its sending units and applying the sigmoid transformation. An activation is computed at each unit and for each input pattern in the training sample.

Let $\mathbf{a}_1$ denote a vector of activations at bias and input units, and let $\mathbf{w}_1$ represent the vector of connection strengths (weights) associated with the connections leading to h_1 (hidden unit 1). Then, the activation for the input pattern at h_1 is obtained by

$$b_1 = f(\mathbf{a}_1'\mathbf{w}_1). \quad (2)$$

Now h_1 as well as the bias and input units send contributions to h_2. Let $\mathbf{a}_2' = [\mathbf{a}_1'|b_1]$, and let $\mathbf{w}_2$ represent the weights associated with connections leading to h_2. The activation at h_2 is then obtained by $b_2 = f(\mathbf{a}_2'\mathbf{w}_2)$. A similar process is repeated until an activation at the output unit is obtained, which is the network prediction for the output. In the training phase, connection weights are determined so that the network prediction closely approximates the output corresponding to the input pattern.

We thus obtain, for each input pattern, a set of activations at different units. The activations (excluding those at the output unit) for all input patterns may be collected in a matrix, $\mathbf{A}$. Activation patterns obtained at each hidden unit constitute a column in $\mathbf{A}$. Let $\mathbf{D}_w$ represent a diagonal matrix of weights leading to the output unit. A matrix of contributions to the output is expressed as

$$\mathbf{Z} = \mathbf{A}\mathbf{D}_w. \quad (3)$$

Let $\mathbf{1}$ denote a vector of ones of appropriate size. Let

$$\mathbf{z} = \mathbf{Z}\mathbf{1} = \mathbf{A}\mathbf{D}_w\mathbf{1} = \mathbf{A}\mathbf{w}, \quad (4)$$

where $\mathbf{w} = \mathbf{D}_w\mathbf{1}$. The sigmoid transformed $\mathbf{z}$ gives the network predictions for $\mathbf{y}$'s. Figure 3 depicts each column of $\mathbf{Z}$, $\mathbf{z}$ and $\mathbf{y}$'s as functions of x_1's and x_2's. This helps us understand visually what the contributions are like, what the summed contributions are like, and how they are turned into network activations by the sigmoid transformation. Note that figures analogous to Figure 3 can be drawn for activations at other units than the output unit.

A similar graphing technique can be used to gain further insights into how a function approximation is done. We may eliminate some of the connections in the net, and see how the function approximation deteriorates. The difference between the original (fully approximated) function and the deteriorated function represents the effect of the eliminated connections. Elimination of some connections may have extremely deteriorating effects in the network performance, but others may not. We see which

connections play important roles, and which not so important. Figure 4 depicts the approximated function when the connection associated with the largest weight (input 1 to h_4) was eliminated. The effect was still relatively minor in this case.

Elimination of a set of connections may entail elimination of all the effects of a unit, only the direct effects of the unit or the indirect effects of the unit. In this fashion we can isolate the total, direct, and indirect effects of a unit in function approximations. Figure 5 depicts how the function approximation changes when all but the direct effect of bias are eliminated, when the direct effects of the two input units are recovered, when the direct effect of h_1 is recovered, and so on.

5 Reduced-Rank Approximation

PCA may be applied to the above $\mathbf{Z}$, and reduced-rank approximations may be obtained. By taking the row sums of the reduced-rank approximations to $\mathbf{Z}$ we obtain the reduced-rank function approximations depicted in Figure 6. The best reduced-rank approximation in terms of network performance is the rank-4 approximation. There is little change in the approximated function beyond rank-4, but the network performance actually deteriorates somewhat. Components beyond 4 are thus considered noise components. Although the network structure (net topology and number of hidden units) typically varies in CC nets depending on initial weights (hereditary differences) for the same task and with the same training sample, the results of reduced-rank approximations seem to have some generality across different nets. The target function for the continuous xor problem was always approximated well with four components, and they correspond with the four terms in the target function: bias, two input units and the interaction between the two input units (i.e., $x_1 \times x_2$).

6 Developmental Data

The foregoing discussion assumes that a net has already been constructed and trained. What about function approximations earlier in the net's history? As noted, the CC algorithm starts as a net without hidden units. It tries to improve its performance within a given net topology by adjusting the connection weights, but as soon as it senses that it can no longer improve its performance by merely adjusting the weights, it changes the net topology by adding a hidden unit. It then readjusts the connection weights. This process is repeated until a prescribed criterion is reached. The topological changes in the net define distinct developmental stages in the training. We can derive $\mathbf{Z}$, $\mathbf{z}$ and $\mathbf{y}$'s (analogous to those defined above) in each developmental stage. Figure 7 depicts function approximations in different stages, and their changes from one stage to the next. Figure 8 depicts the role history of the bias unit over the developmental stages. Similar figures can be drawn for other units.

7 PCA, CPCA and PARAFAC

Plotting component scores obtained in PCA (Principal Component Analysis) of $\mathbf{Z}$ are often informative for characterizing the nature of components (Shultz, Oshima-Takane & Takane, 1994). See Figure 9. Also, $\mathbf{Z}$ may contain known components. For example, it contains linear effects of bias and two input units. These known effects may be eliminated before PCA is applied to the residuals. This highlights more interesting aspects of $\mathbf{Z}$, such as interactions among input variables (Constrained PCA or CPCA; Takane & Shibayama, 1991; see Figure 10). PARAFAC may also be interesting when there are more than one output unit. The matrix of contributions, $\mathbf{Z}_k$, to output unit k is expressed as

$$\mathbf{Z}_k = \mathbf{A}\mathbf{D}_{wk}, \tag{5}$$

where the matrix of activations, $\mathbf{A}$, is common to all k. PARAFAC (Harshman, 1972) decomposes each $\mathbf{Z}_k$ by

$$\mathbf{Z}_k = \mathbf{U}\mathbf{D}_k\mathbf{V}', \tag{6}$$

where $\mathbf{D}_k$ is diagonal and specific to k, but $\mathbf{U}$ and $\mathbf{V}$ are common to all k. The $\mathbf{U}$ (matrix of component scores) in PARAFAC is not rotatable, so that more substantive meanings may be attached to the components obtained by PARAFAC. The same technique can also be used for reduced-rank approximations for developmental data. In this case, the subscript k refers to a developmental stage. The diagonal elements of $\mathbf{D}_k$ indicate how much of various components enter into functions approximated in different stages.

8 Environmental Bias

The approximated function obtained by a net typically varies with training sample. If a random sample of possible training patterns are used, we obtain a less biased function approximation than when the training sample represents a biased subset of possible patterns. For example, what happens if all the training patterns used are far away from, or all close to the boundaries of the two classes of input patterns to be discriminated in the continuous xor problem. It is likely that we obtain sharper boundaries in the latter. Investigating the effects of environmental bias is important at least in two respects. It is the long standing problem in statistics to find an optimal set of training patterns for a particular task, and changes in environment are considered a major source of changes in behavior. Dramatic effects of environmental bias were found in acquisition of personal pronouns (Oshima-Takane, 1988, 1990; Shultz et al., 1994) as well as in a number of other learning situations.

9 Generalizations

So far we have discussed function approximations only for the training patterns, whereas the functions to be approximated often go beyond the domain of the training patterns used in the training. This is called the generalization problem. Generalization includes both interpolation and extrapolation. In the continuous xor problem, for example, what happens to the function approximation at $x_1 = 0.38$ and $x_2 = 0.73$ (interpolation) and what about at $x_1 = -0.1$ and $x_2 = 1.2$ (extrapolation) which was never used in the training? We investigate the generalization capability of the CC nets using techniques similar to the above.

10 Other Tasks

There are other interesting tasks than the continuous xor problem. Similar analyses can be performed on simulation data obtained from these tasks. Some of the possible tasks are:

(a) The addition-multiplication problem. This problem has four input variables. Both x_1 and x_2 range from 1 to 9, and x_3 from 1 to 82. The x_4 designates whether the operation applied to x_1 and x_2 is addition ($x_4 = 0$) or multiplication ($x_4 = 1$). Let u denote the result of an operation applied to x_1 and x_2 ($u = x_1 + x_2$ if $x_4 = 0$, and $u = x_1 \times x_2$ if $x_4 = 1$). The problem is to judge if u is greater than, smaller than, or equal to x_3.

(b) The two interlocking spirals problem.

(c) Discrimination of odd and even numbers.

(d) Pronoun learning. When a mother talks to her child, me refers to herself, and you to the child. However, when the child talks to the mother, me refers to the child, and you to the mother. How children learn this reversal has extensively been studied by Oshima-Takane and her collaborators (Oshima-Takane, 1988, 1990; Shultz et al., 1994). The problem can be regarded as a type of concept learning, where the concepts (or rules) to be learned are: use me when the speaker and the referent agree, and use you when the listener and the referent agree.

These tasks differ in difficulty, but all have well-defined structures.

11 References

Fahlman, S. E., & Lebiere, C. (1990). The cascade correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524-532). San Mateo, CA: Morgan Kaufmann.

Harshman, R. A. (1972). Foundations of parafac procedure: Models and conditions for an "exploratory" multi-modal factor analysis. UCLA Phonetic Lab Working Paper # 16.

Hornik, M., Stinchcombe, M., & White, H. (1989). Multilayer feed-forward networks are universal approximators. *Neural Networks*, **2**, 359-366.

Oshima-Takane, Y. (1988). Children learn from speech not addressed to them: The case of personal pronouns. *Journal of Child Language*, **15**, 94-108.

Oshima-Takane, Y. (1990). Analysis of pronomial errors: A case study. *Journal of Child Language*, **19**, 111-131.

Oshima-Takane, Y., Goodz, E., & Derevensky, J. L. (in press). Birth order effects on early language development: Do second born children learn from overheard speech? *Child Development.*

Ripley, B.D. (1993). Statistical aspects of neural networks. In O. E. Barndorff-Nielsen, J. L. Jensen, & W. S. Kendall (Eds.), *Network and chaos — Statistical and probabilistic aspects* (pp. 40-123). London: Chapman and Hall.

Shultz, T. R., Buckingham, D., & Oshima-Takane, Y. (1994). A connectionist model of the learning of personal pronouns in English. In S. J. Hanson, T. Petsche, M. Kearns, & R. L. Rivest (Eds.), *Computational learning theory and natural learning systems, Vol. 2: Intersection between theory and experiment* (pp. 347-362). Cambridge, MA: MIT Press.

Shultz, T. R., Oshima-Takane, Y., & Takane, Y. (1994). Analysis of unstandardized contributions in cross connected networks. Paper accepted by NIPS.

Shultz, T. R. et al. (1994). Modeling cognitive development with a generative connectionist algorithm. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling.* Hillsdale, NJ: Erlbaum.

Takane, Y., & Shibayama, T. (1991). Principal component analysis with external information on both subjects and variables. *Psychometrika*, **56**, 97-120.

Takane, Y., Oshima-Takane, Y., & Shultz, T. R. (1994). Methods for analyzing internal representations of artificial neural networks. In T. Kubo (Ed.), *The Proceedings of the 22nd Annual Meeting of the Behaviormetric Society* (pp. 246-247). Tokyo: The Behaviormetric Society.

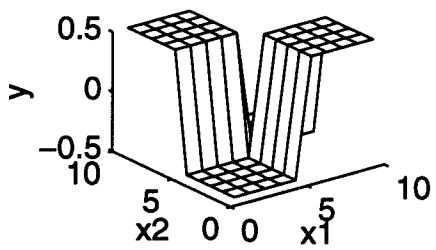

Figure 1: The target function for the continuous xor problem.

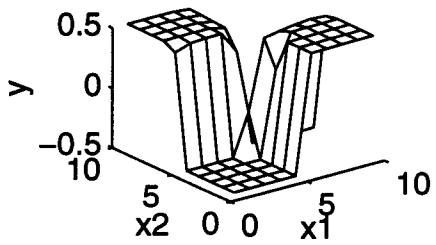

Figure 2: A net approximation to the target function.

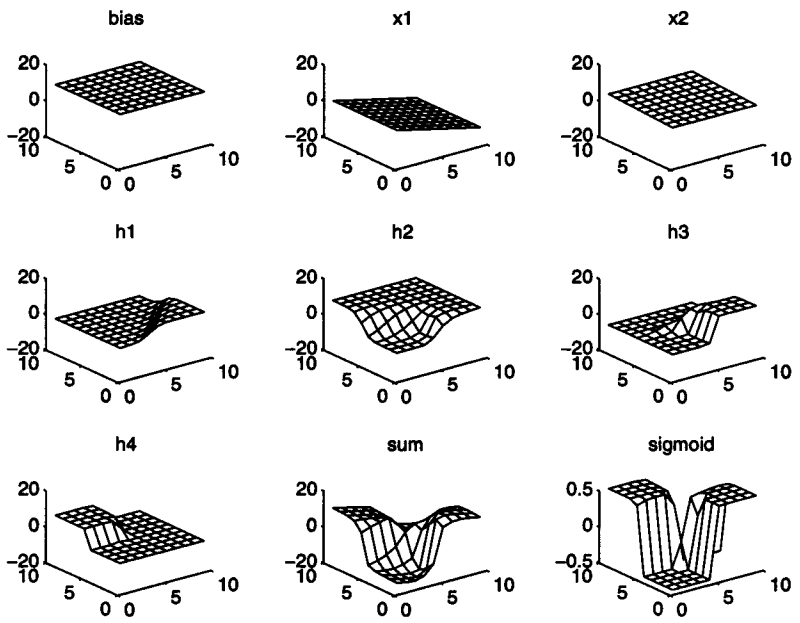

Figure 3: Contributions of units and network predictions for the output unit.

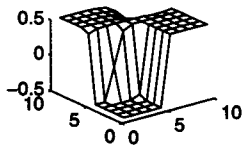

Figure 4: The deteriorated function.

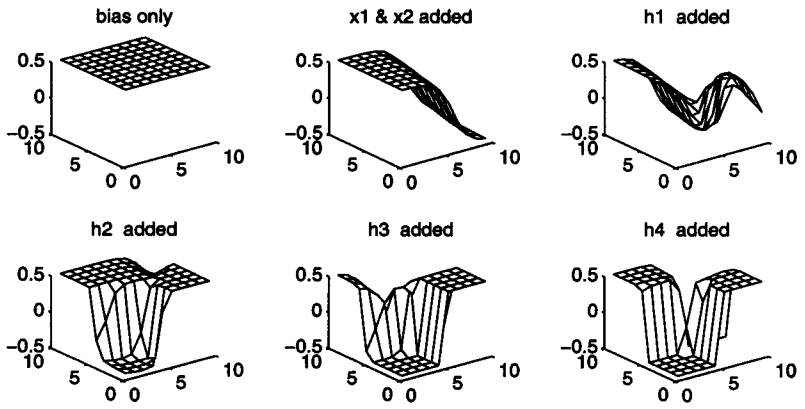

Figure 5: Function approximations with partial connections.

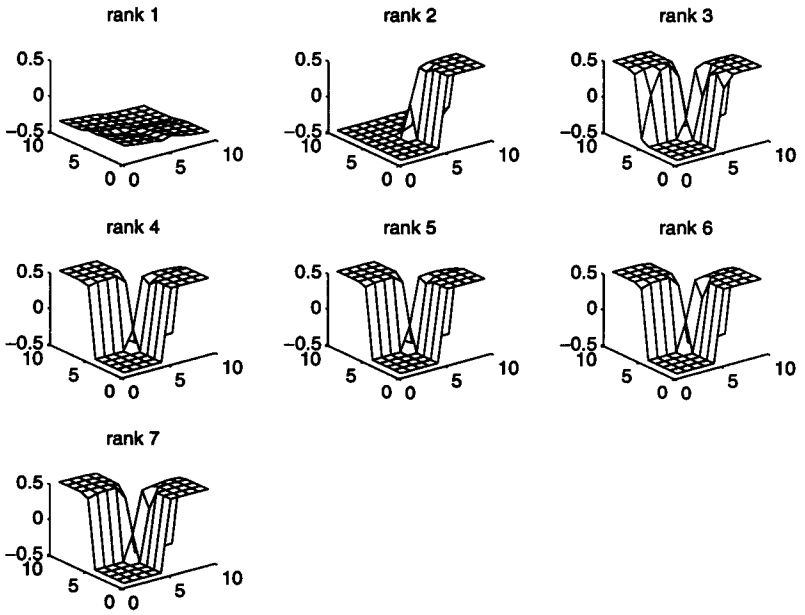

Figure 6: Reduced-rank approximation.

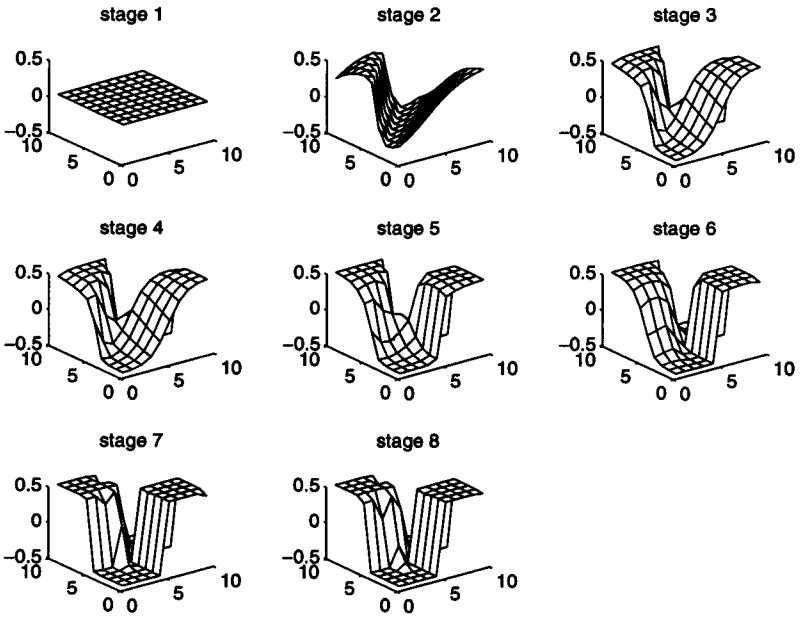

Figure 7: Changes in function approximations.

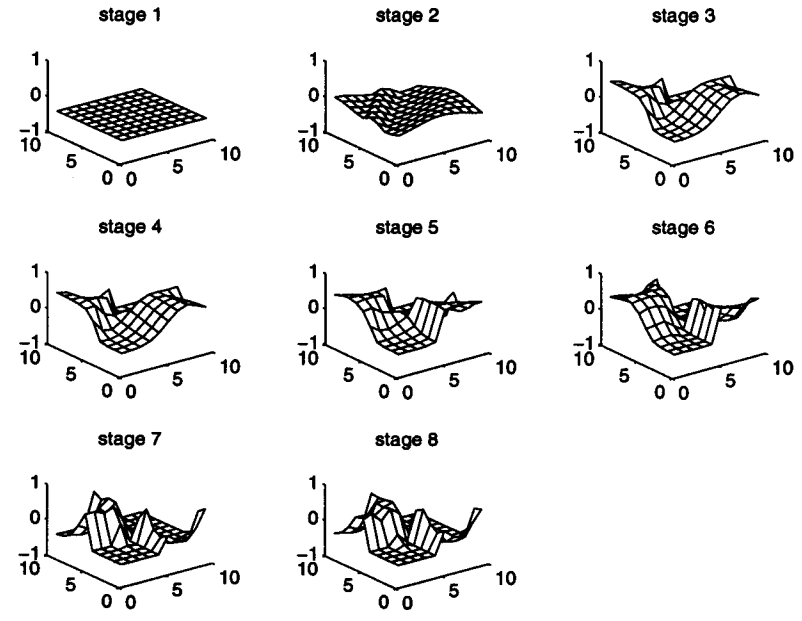

Figure 8: Role history of the bias unit.

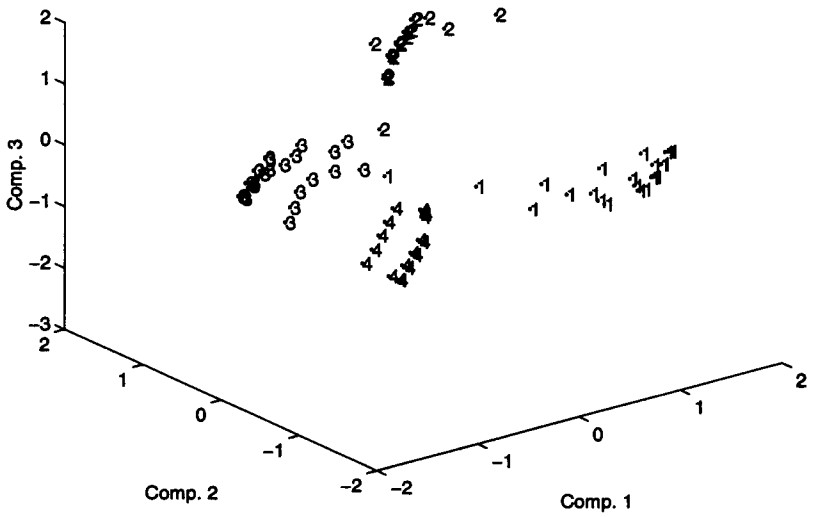

Figure 9: Plot of component scores from PCA.

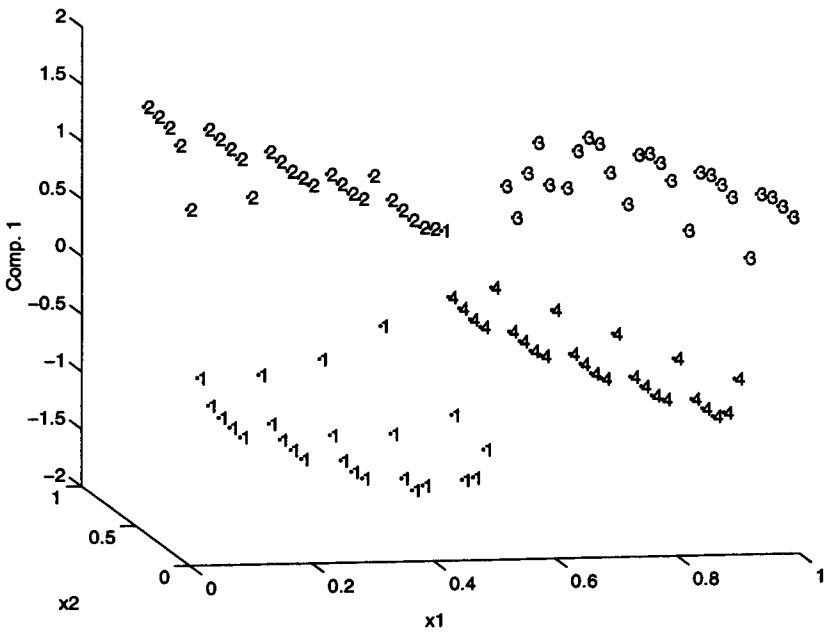

Figure 10: Plot of the first component from CPCA.