This paper presents randomization of input weights followed by closed-form solution by pseudo-inverse (the same as Moore-Penrose generalized inverse) for output weights on page 167. There are several follow-up works in this direction. All these were excluded in the ELM-SLFN paper in 2004 (PDF: Huang IJCNN 2004)

Neurocomputing 6 (1994) 163–180
Elsevier



NEUCOM 269

The only difference between ELM-SLFN and RVFL is that RVFL has direct links from the input to the outputs. This removal makes ELM-SLFN worse than RVFL.

本文在第167页给出了输入权值的随机化和输出权值的伪逆(与Moore-Penrose广义逆相同)的闭合解。在这个方向上有一些后续的工作。所有这些在ELM-SLFN 2004年的论文中都被排除在外(PDF: Huang IJCNN 2004)

# Learning and generalization characteristics of the random vector Functional-link net

ELM-SLFN和RVFL之间的唯一区别是RVFL有从输入到输出的直接链接。这种去除使得榆树-slfn比RVFL更糟糕。

**Yoh-Han Pao*, Gwang-Hoon Park and Dejan J. Sobajic**

*Electrical Engineering and Applied Physics, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106-7221, USA*

Received 3 December 1992
Revised 12 July 1993

论文给出了在一个输出维度下的RVFLNN和GDR net在学习一个已知的函数和真实的现实数据下两种学习情况的对比，并在论文后半部分给出RVFLNN在单输出维度下的数学证明。拟合函数部分：GDR net在训练迭代次数不足的时候会出现严重的拟合偏差Fig9。拟合真实数据部分：过度训练GDR net会出现过拟合而RVFLNN则只会出现轻微的偏差。对于RVFLNN增加增强节点会显著减少训练误差Fig10，而且所需数量不会太多。由单个输出维度可以简单推广到多个维度论文页码177红色字体块

*Abstract*

In this paper we explore and discuss the learning and generalization characteristics of the random vector version of the Functional-link net and compare these with those attainable with the GDR algorithm. This is done for a well-behaved deterministic function and for real-world data. It seems that '*overtraining*' occurs for stochastic mappings. Otherwise there is saturation of training.

*Keywords*. Neural net; Functional-link net; functional mapping; generalized delta rule; auto-enhancement; overtraining and generalization.

## 1. Introduction

Of all the capabilities ascribable to neural-net computing, there is none more noteworthy than that of *supervised learning*. The work of Hornik, Stinchcombe and White [1], Funahashi [2] and Cardaliaguet and Euvrard [3] indicate that the multi-layer generalized *PERCEPTRON* feedforward net with linear links and appropriate nonlinear activation at nodes can serve as a computational model of functional mappings $f(x)$ from $R^N$ to $R$. Such existence theorems do not address issues such as learning and generalization, and different approaches to those matters can be adopted with correspondingly different results in practice.

In the past, we have explored variations [4, 5] on the feedforward net architecture, inspired primarily by the work of Giles and Maxwell [13] on high-order neural networks. Our idea is that often it might be helpful to abstract and replace substantial parts of an otherwise massive net with use of Functional-links. Functional-link (FL) nets can be implemented in various ways, one of which is the random vector (RV) approach [7, 8]. This paper is concerned with the learning and generalization characteristics of the random-vector implementation of the FL net. These are described and discussed comparatively with corresponding characteristics of the backpropagation-of-error (BP) [9] net for two types of situations; one for which a

* *Corresponding author*. Fax: 1 216 368 2668.

deterministic causal mapping does exist and another involving *real world* noisy data. In this work the BP net is also referred to interchangeably as the generalized delta rule (GDR) net.

In the neural-network supervised learning task, the generic idea is that such a net can synthesize a network computational model of a known function *if that function does indeed exist*. The learning procedure is based on a finite number of known instances of the *presumed functional mapping*, and it is assumed and hoped that the computational network model so learned is not only capable of replicating the known instances of the mapping but is also valid for the infinitude of all the other points in the neighborhoods of the exemplars.

In practice many things can go awry. For example, there may be, in fact, no deterministic function and the exemplars might be merely a set of random pairings of points. In such a case the training set of mappings can still be learned very well, but any test set will exhibit the random scatter inherent to the stochastic process used to generate the data. If a net does not have enough available adjustable parameters or if the learning process is terminated prematurely, then both the training set and test set errors may be large. On the other hand if the net has too many adjustable parameters or if the net is *over-trained*, then the training set error may be deceptively low without comparable performance attainable for the test set points.

## 2. Brief review of the random vector Functional-link approach

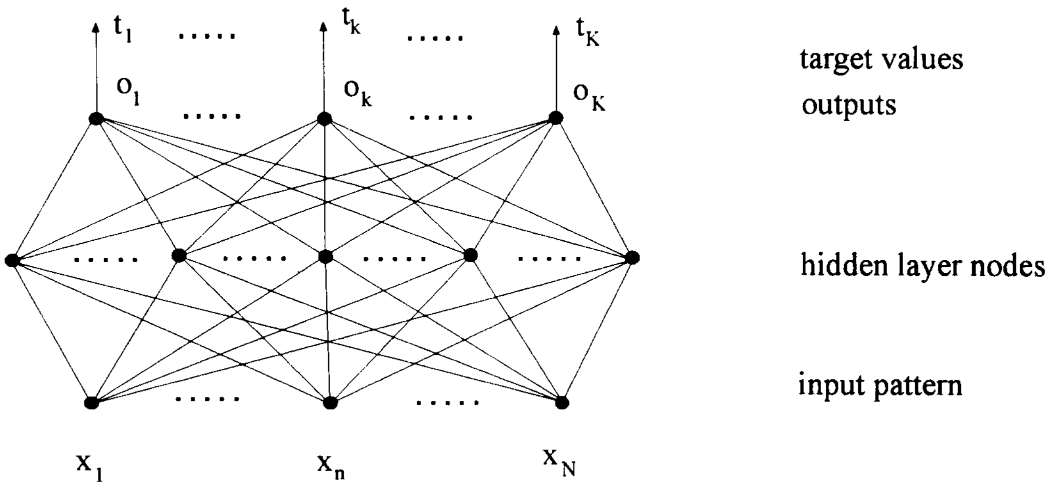
The basic GDR concept is illustrated in *Fig. 1(a)* for a mapping from an $N$-dimensional space to $K$-dimensional space. There are $J$ nodes in the hidden layer and the action of the net may be understood in terms of two successive mappings or transformations. The action of the initial mapping, from the input to the hidden layer, is to transform the description of the input pattern vectors from the original one in input space into another description in *internal representation* space. In that new representation, the next mapping is a linear one. In the GDR or BP approach, all the network parameters are successively adjusted until the known mappings are replicated to the desired accuracy for all the vector parts provided as the *training set*.

The random-vector FL network shown in *Fig. 1(b)* performs a nonlinear transformation of the input pattern before it is fed to the input layer of the network. The essential action is, therefore, the generation of an enhanced pattern to be used in place of the original. Experience indicates that supervised learning can be achieved readily using a *flat* net (one that has no hidden layers), and that the delta rule can be used in place of the generalized delta rule (GDR) if this enhancement is done correctly.

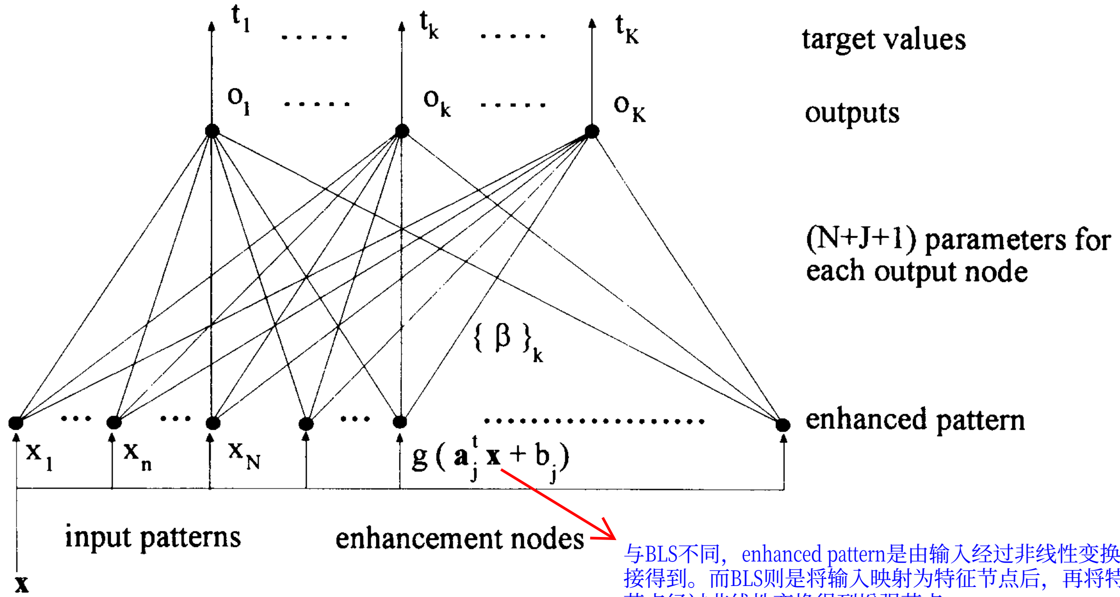
The random-vector FL net is but one specific mode of realization of the general Functional-link net idea. Other modes of instantiations of that approach have been described and discussed by us elsewhere in previous publications [4–7], but the random-vector version is attractive because it is susceptible to rigorous mathematical proof [14] and also because it is easy to use. In connection with the latter comment, we note that even though a large number of enhancement nodes might be generated initially, usually a large fraction of those candidate enhancement nodes can be pruned away if they, individually, do not contribute to the net input of the output nodes, or sometimes, depending on the circumstance, if a node does not contribute to discrimination between classes.

The network connectivities shown in *Fig. 2(a)* and *(b)* are similar to those of *Fig. 1*, except for the fact that we focus on a single output and do not require a nonlinear transform at that

图2和图1相比，图2专注与一个输出，因为对于一个输出
的进行数学推导同样适用于其他输出，本质是完全一样的




(a) the GDR net




与BLS不同，enhanced pattern是由输入经过非线性变换直接得到。而BLS则是将输入映射为特征节点后，再将特征节点经过非线性变换得到增强节点。

(b) the Random-Vector FL net

Fig. 1. Comparison of network connectivities.

final output. This allows us to illustrate explicitly some of the entities to be learned and there is no loss of generality.

To be specific for the GDR net, the input to each hidden layer node is $\text{net}_j = \sum a_{jn} x_n =$

看图2来理解

图2相比图1专注于单个输出而不需要在最终输出处进行非线性变换。

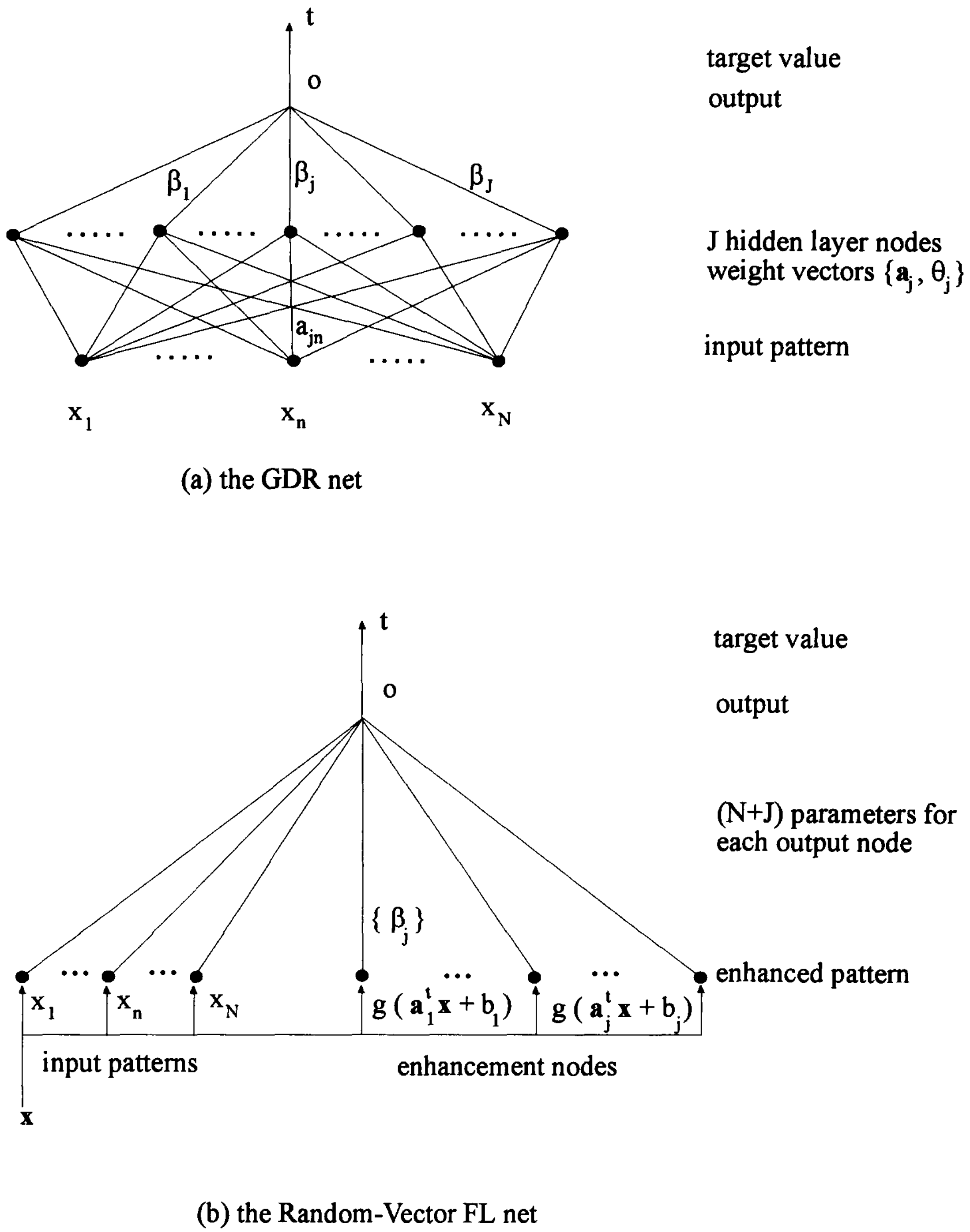



Fig. 2. Illustration of network parameters.

$a_j^t x$ and the output is $g(a_j^t x + b_j)$ where $b_j$ is the threshold parameter for the node $j$ and $g(\cdot)$ is the sigmoid function. The net to the single (nonlinear) output is simply $\sum \beta_j g(a_j^t x + b_j)$. In the *random-vector* FL net, the functional enhancements are achieved in essentially the same

RVFLNN的增强节点实现，就跟GDR网络的隐藏层的实现是一样的。

原始输入到增强节点的权重aj是随机产生的，但是向量aj需要受到约束，以使激活函数g(a)x + bj)在大多数情况下不饱和。

FL网络的随机向量实现可以被视为与GDR本质上相同，除了隐藏层被向下移动以作为输入向量的增强，权重向量{aj}不是学习的，而是随机生成的(但适当地)。

manner as in the first layer of the GDR net. The additional enhancements are $g(\boldsymbol{a}_j^t \boldsymbol{x} + b_j)$. The random-vector implementation of the FL net can be viewed as essentially the same as the GDR except for the fact that the hidden layer is moved down to serve as an enhancement of the input vector *and* the weights vectors $\{\boldsymbol{a}_j\}$ are *not* learned but are randomly generated (but appropriately[1]). The random-vector implementation can also be viewed as a type of encoding comparable to the coarse-coding of patterns with linguistic symbolic feature values [10]. A mapping from $N$-dimensional space to $K$-dimensional space is represented by $K$ independent networks and each network can be modified adaptively without entanglement with the others.

## 3. Learning and generalization characteristics

N个输入节点x，和由输入节点非线性变换得到的J个增强节点

For the random-vector FL net, only the weights $\beta_j$ need to be learned. Including the original inputs, there are $(N+J)$ components in the enhanced pattern and there are accordingly $(N+J)$ weights (or $\beta_j$ values), to be determined.

每个节点到一个最终输出节点都需要一个权重总共N+J个

Learning is by minimization of the system error defined as

看下面的real-world，这个patterns又像是输入数据实例的个数

$$E = \frac{1}{2P} \sum_{p=1}^{P} (t^{(p)} - \boldsymbol{B}^t \boldsymbol{d}^{(p)})^2$$

t就是输出值

这个enhanced pattern应该包括输入input pattern的节点和enhancement nodes吧，总共N+J个节点，这样能B的维度匹配

where $\boldsymbol{B}^t$ is the vector of weight values $\beta_j$, $j = 1, 2, 3, \ldots, N+J$, and $\boldsymbol{d}$ is the enhanced pattern vector (not the original input pattern vector). There are $P$ training set patterns and the subscript $(p)$ is the pattern index.

p应该指的是模型最后的维度，也就是输出维度，弄了一个training set pattern概念

$E$ is quadratic with respect to the $(N+J)$ dimensional vector $\boldsymbol{B}$. This means that the unique minimum can be found in no more than $N+J$ iterations of a learning procedure such as the conjugate gradient (CG) approach [11, 12], if the explicit matrix inversion needs to be avoided. If matrix inversion with use of a pseudo-inverse is feasible, then a single step learning would suffice.

使用共轭梯度来求得最优的B，如果直接能求伪逆就更简单了

Several publications prior to 2004 have used randomization with closed-form solution as listed by Wang and Wan (PDF: Wang Huang TNN 2008).

In this paper, we report on the learning and generalization characteristics of the random-vector FL net for two different circumstances, one being that of a deterministic $R^1$ to $R^1$ mapping and the other being a mapping for noisy *real-world* data.

A well-behaved continuous function might be that shown as $f_1(x)$ in *Fig. 3*. However, the circumstance is that the function $f_1(x)$ is known to us only through the point marked with bold dots. As we will describe in further detail in the following, the function $f_2(x)$ is another function, learned in error by the GDR net if training is terminated prematurely. As shown in *Fig. 4*, the FL net can achieve satisfactory learning of the training set points if 300 enhancements are used. A system error of 0.000025 is attained after 12,065 iterations. The net is a flat net, i.e. a linear net. We did not use the CG approach partly because we want the FL and GDR net results to be comparable and also because of simplicity in the straightforward gradient iterative approach.

In generalization mode the performance of the FL net is not bad, as illustrated in *Fig. 5*. Comparable learning is achieved by a GDR net with 20 hidden layer nodes. The accuracy

---

[1] The vector $\mathbf{a}_j^t$ needs to be constrained so that activation functions $g(\mathbf{a}_j^t \mathbf{x} + b_j)$ are not saturated most of the time.

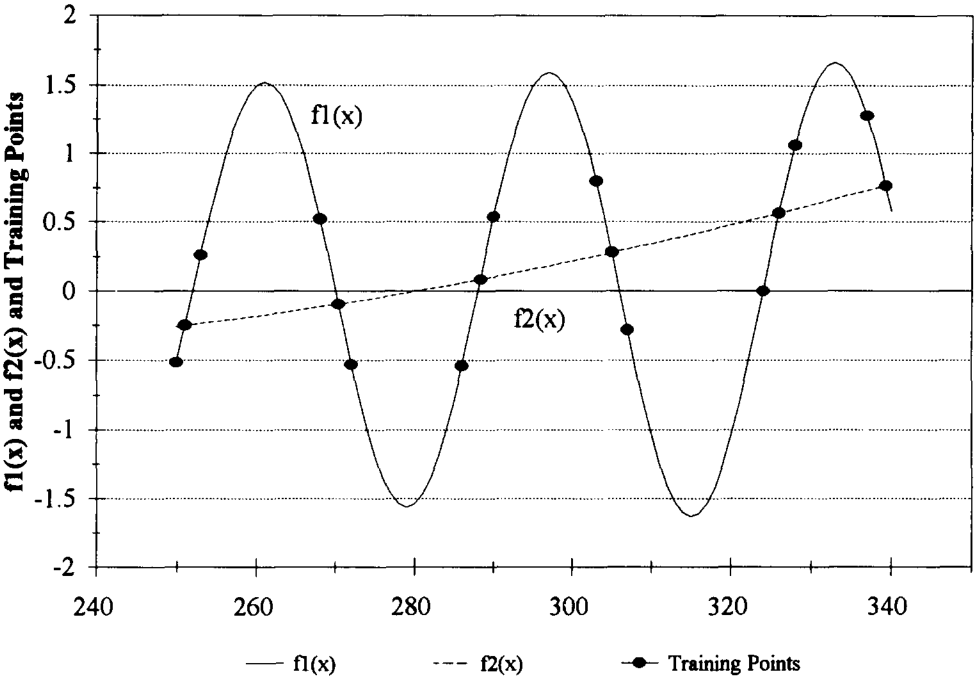



Fig. 3. The function $f_1(x)$ to be learned, the training set points and the erroneous function $f_2(x)$.

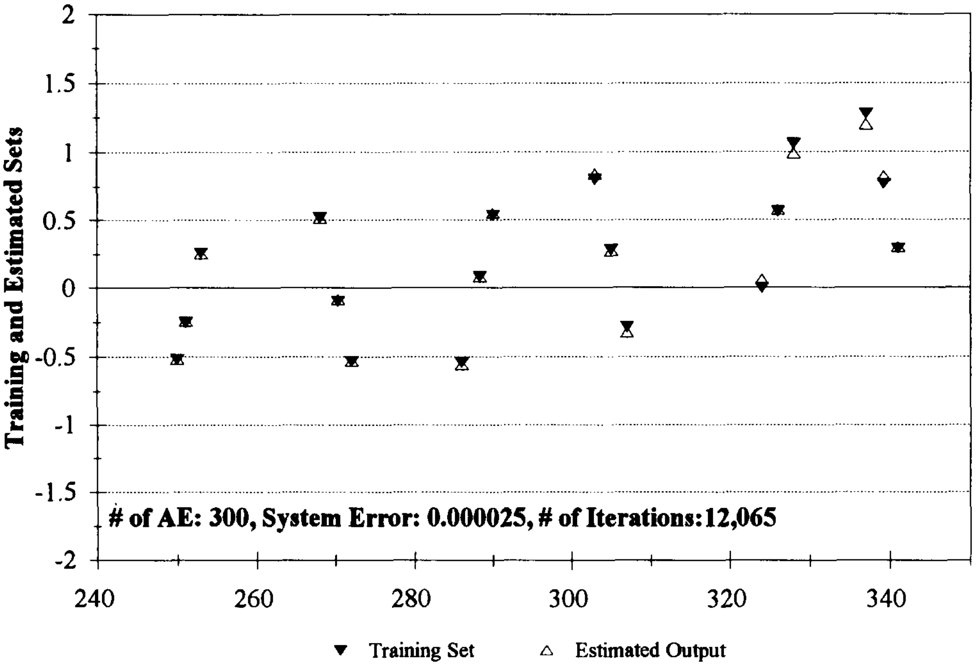



Fig. 4. Learning achieved by FL net for training set points.

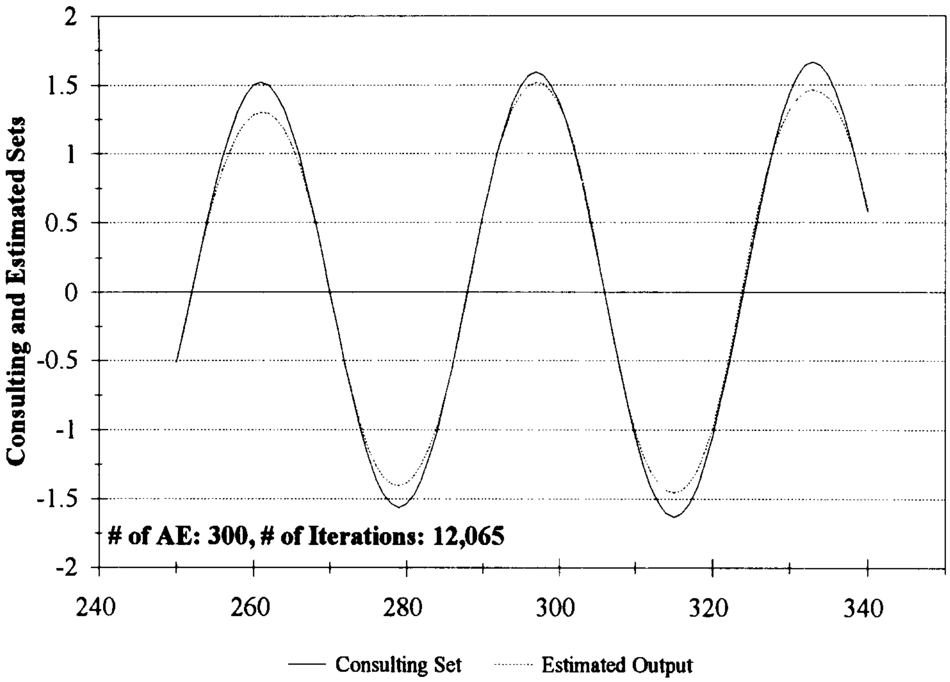



Fig. 5. Comparison of actual and estimated function values for FL net in generalization mode.

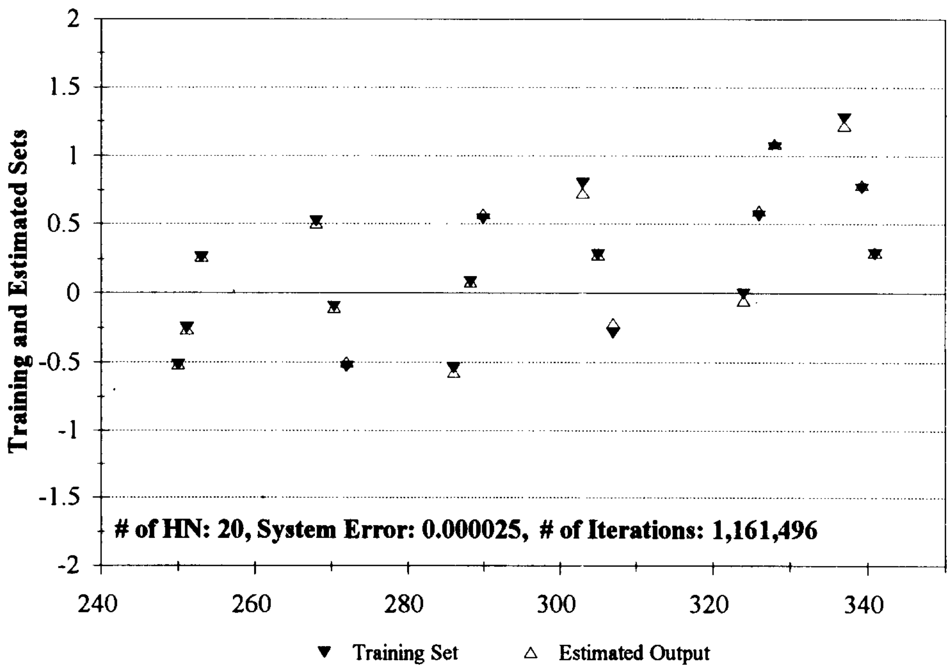



Fig. 6. Learning achieved by GDR net for training set points.

of the learning achieved by the GDR net for the training set points is illustrated in *Fig. 6*, but this was achieved only after 1,161,496 iterations. In generalization mode the performance of the GDR net is not bad either, as illustrated in *Fig. 7*. It is interesting to note that if GDR learning were stopped at 200,000 iterations, the function learned is $f_2(x)$ and not $f_1(x)$. This is shown for the training set points and for the generalization mode in *Fig. 8* and *Fig. 9* respectively.

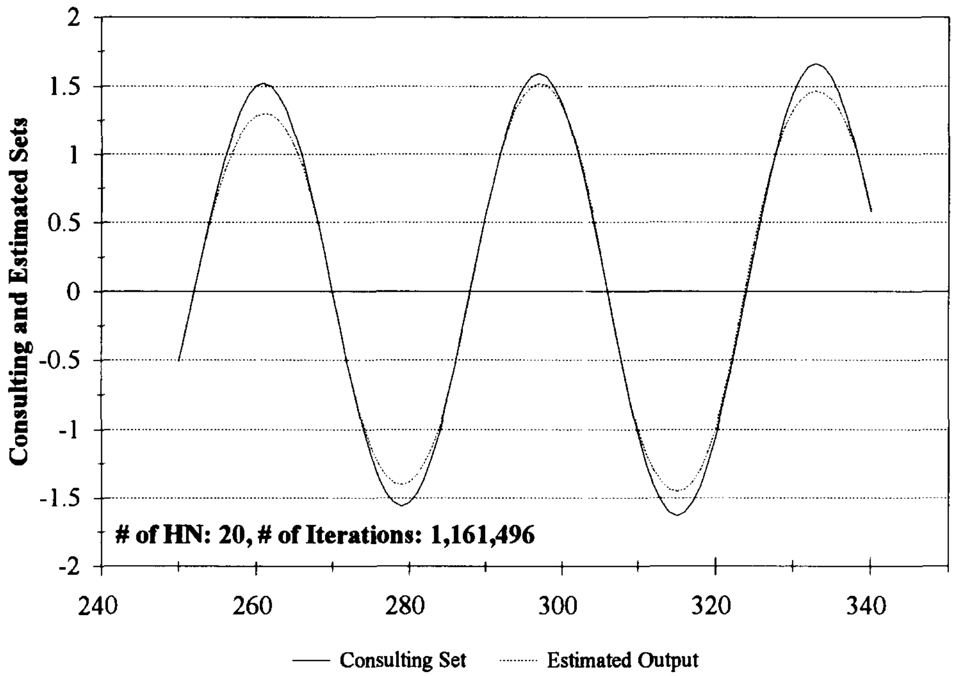



Fig. 7. Comparison of actual and estimated function values for GDR net in consulting mode.

FL网的随机向量版本的特点是需要相对大量的自动增强，即额外的增强节点。这是不值得反对的，因为所需的数量似乎并没有随着所涉及模式的维度而增加多少。

The random vector version of the FL net is characterized by the need for a relatively large number of *auto-enhancement*, i.e. the additional enhancement nodes. This is not objectionable because the number needed does not seem to grow much with the dimension of the patterns involved.

*Figure 10* shows that the number of iterations required to achieve a system error 0.000025 decreases rapidly with the increase in the number of auto-enhancements. The difference between the performances achievable with the training and test set is well illustrated by *Figs. 11* and *12*. From the curves shown in *Fig. 11*, we see that after about 20,000 iterations there is no further significant improvement in the performance of the FL net in so far as generalization is concerned. In this case, it is probably not quite accurate to say that such a net is *overtrained*, but additional improvement is marginal. Similar behavior is shown in *Fig. 12* for the GDR net.

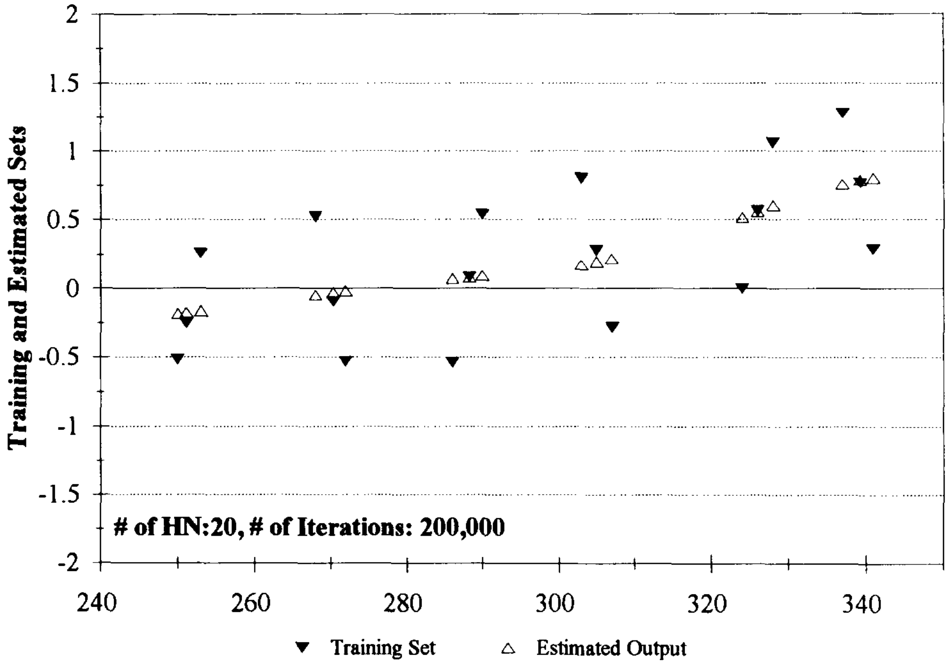



Fig. 8. Failure of the GDR net to learn the training set points if training is stopped at 200,000 iterations.

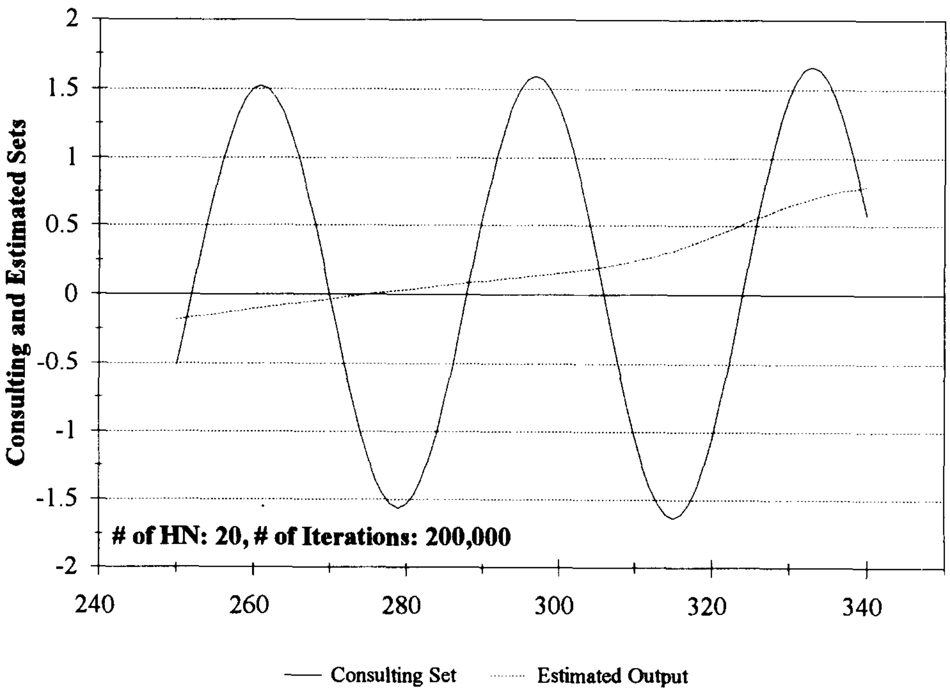



Fig. 9. Failure of 'undertrained' GDR net to generalize adequately in consulting mode.

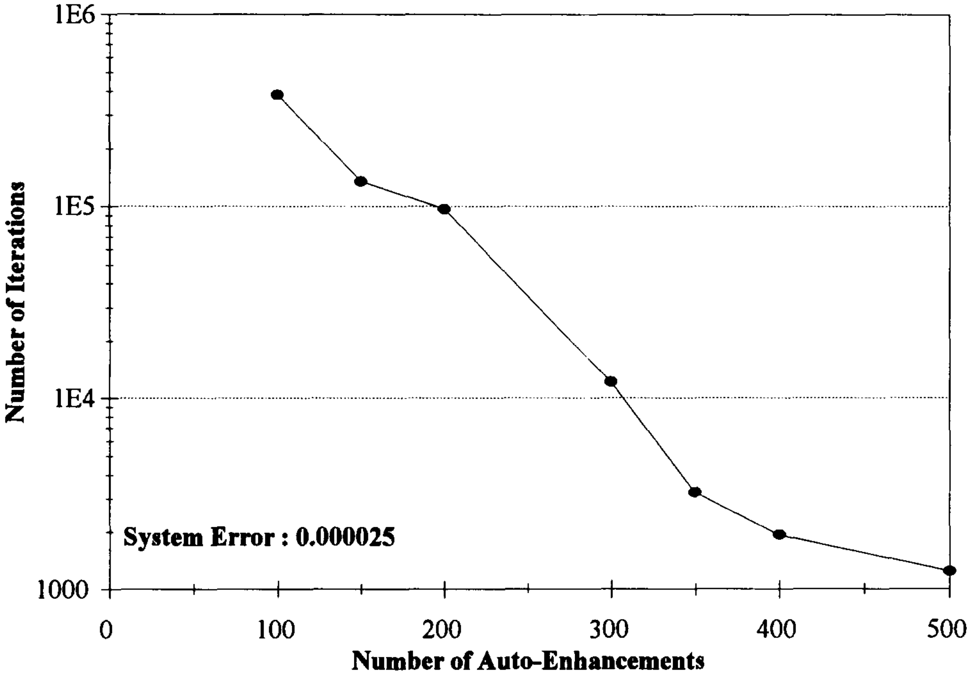



Fig. 10. Variation of number of iterations needed for training with number of auto-enhancements (for FL net).

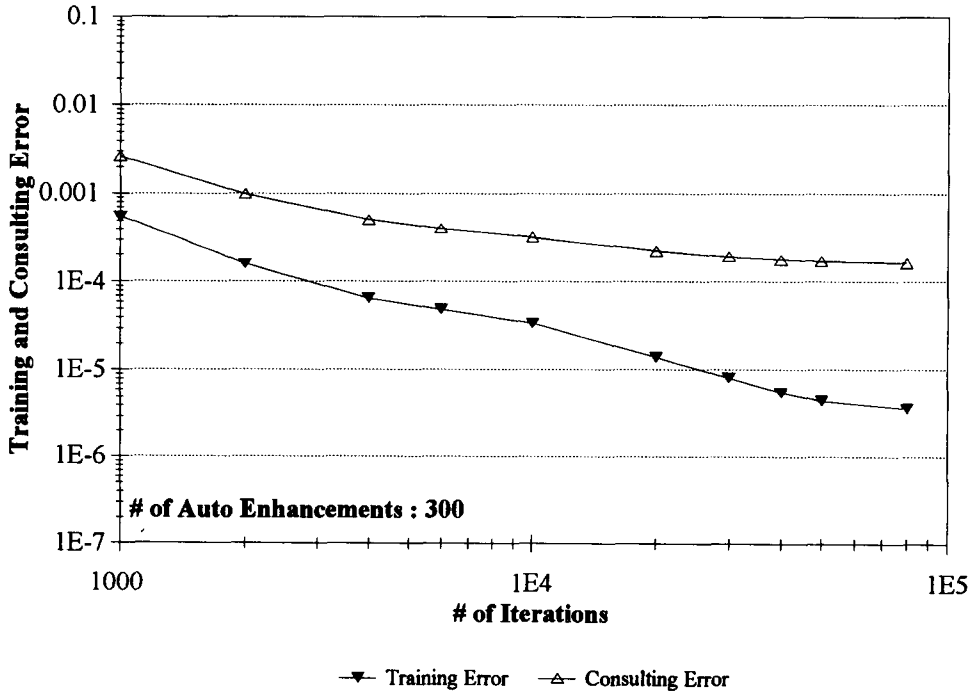



Fig. 11. Saturation of training for the FL net.

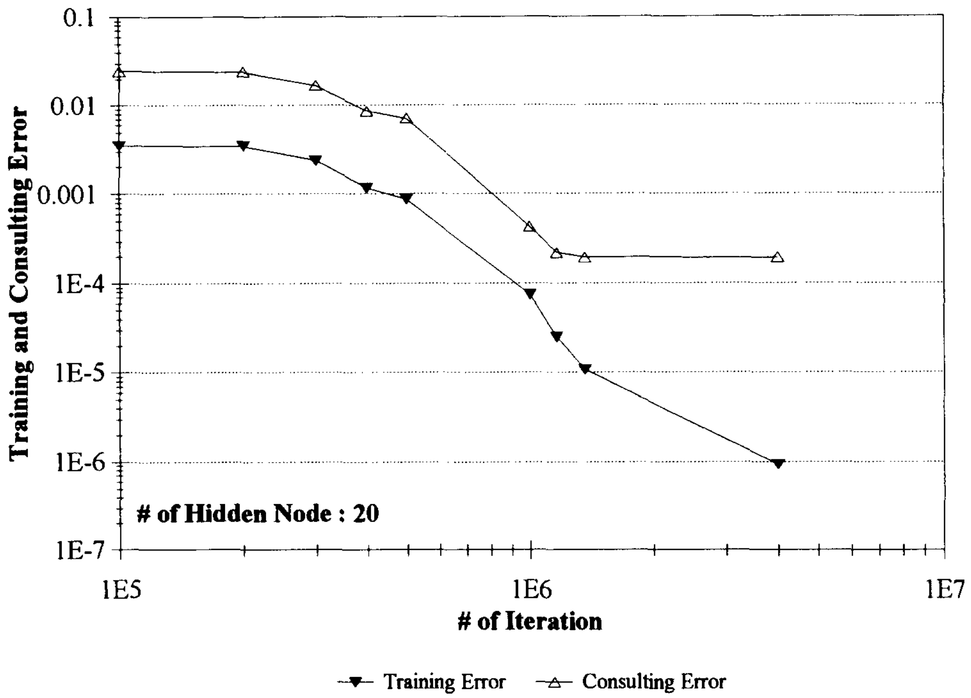



Fig. 12. Saturation of training for the GDR net.

RVFLNN和GDR对于单个已知函数(尽管网络并不知道)。在这种情况下，我们没有出现训练过度的现象，只有训练饱和的现象。在不提高训练集质量的情况下，泛化模式下估计误差的幅度无法进一步减小。

In the preceeding discussion, we have endeavored to present a picture of the learning and generalization capabilities of the random-vector FL net as compared with those of the GDR net for a single well-behaved function which exists and is known to us, even though not known to the nets. Under such circumstances, we do not have the phenomenon of *overtraining* but only that of saturation of training. The magnitude of estimation errors in generalization mode cannot be reduced further without improving the quality of the training set.

对于真实世界的数据，我们可能会进行过度训练，当我们增加训练程度时，实际上会恶化估计值的准确性。GDR比较严重，RVFLNN比较轻微

The situation is different for *real-world* data where the functional mapping to be learned is stochastic. For those cases we can have *overtraining* and actually worsen the accuracy of estimated values as we increase the degree of training.

We illustrate some aspects of such situations with use of one instance of a real-world learning task. For that real-world task, one aspect of the overall training task could be viewed as learning a mapping from $R^4$ to $R^1$. The training set consisted of 28 input patterns with corresponding outputs. These are listed in *Table 1*. The test set of 10 patterns is also described in *Table 1*.

The training and generalization capabilities achieved by the GDR and FL nets are illustrated in *Figs. 13* and *14* respectively. In *Fig. 13*, we see that for the FL net, there is not much point in going beyond about 500 iterations. There is indeed a slight deterioration in consulting capability as we overtrain but the deterioration is minor.

In the case of the GDR net, it is encouraging to note that training should probably be stopped after 1000 iterations, at which point the generalization quality is about twice as good as that of the FL net. But if this is not monitored, then the performance of the generalization mode deteriorates due to overtraining and the performance becomes about only half as good as that of the FL net.

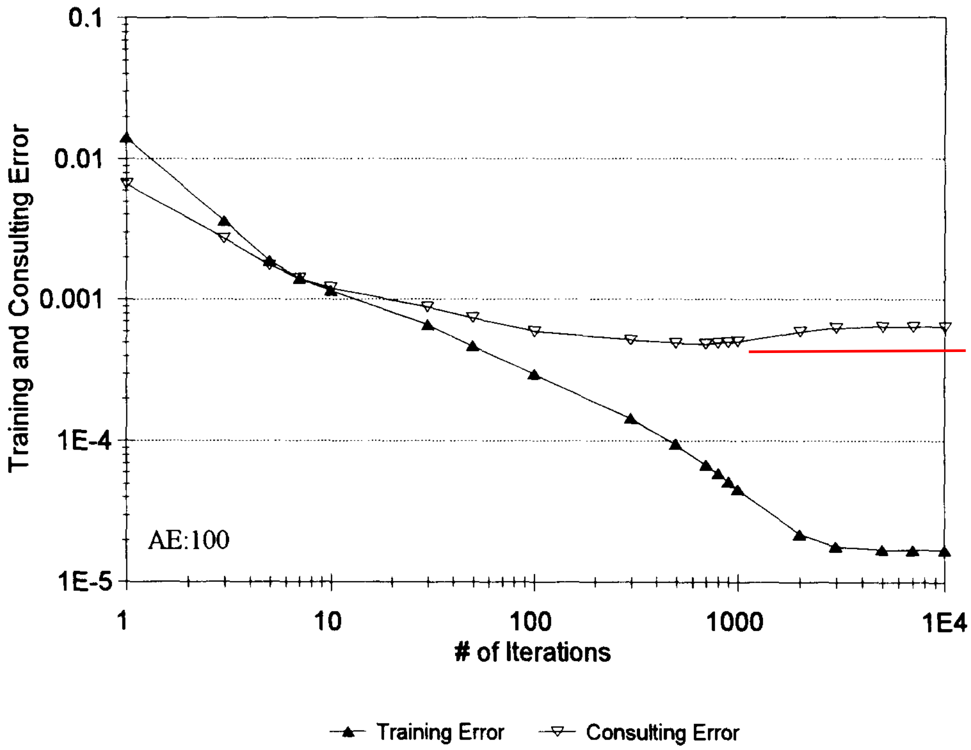
Table 1. Training and consulting sets (patterns are obtained from practical chemical data).

Training Set

| # | X1 | X2 | X3 | X4 | Y |
|---|---|---|---|---|---|
| 1 | 1.25 | 1 | 1.25 | 1.25 | 23.2 |
| 2 | 2 | 1.75 | 3 | 1.75 | 17.4 |
| 3 | 2 | 1.75 | 5.16 | 1.75 | 13.7 |
| 4 | 2 | 1.75 | 0.84 | 1.75 | 24.5 |
| 5 | 2 | 1.75 | 3 | 3.05 | 22.7 |
| 6 | 2 | 1.75 | 3 | 0.45 | 7.8 |
| 7 | 2 | 3.05 | 3 | 1.75 | 16.5 |
| 8 | 2 | 1.75 | 3 | 1.75 | 17.3 |
| 9 | 2 | 1 | 1.75 | 2.5 | 22.6 |
| 10 | 2 | 1 | 3.75 | 1 | 11.7 |
| 11 | 2 | 2.5 | 1.75 | 1 | 15.4 |
| 12 | 2 | 2.5 | 1.75 | 2.5 | 22.7 |
| 13 | 2 | 2.5 | 3.75 | 2.5 | 18.3 |
| 14 | 2 | 2.5 | 3.75 | 1 | 10.7 |
| 15 | 2 | 0.45 | 3 | 1.75 | 16.2 |
| 16 | 2 | 1.75 | 3 | 1.75 | 16.2 |
| 17 | 1.75 | 2 | 2.75 | 2 | 18.5 |
| 18 | 1.75 | 1 | 2.75 | 1 | 15 |
| 19 | 0.75 | 2 | 2.75 | 1 | 17.7 |
| 20 | 0.75 | 1 | 2.75 | 2 | 25.5 |
| 21 | 1.25 | 1.5 | 2 | 1.5 | 20 |
| 22 | 2.25 | 1.5 | 2 | 1.5 | 16.6 |
| 23 | 0.25 | 1.5 | 2 | 1.5 | 33.8 |
| 24 | 1.25 | 0.5 | 2 | 1.5 | 23.5 |
| 25 | 1.25 | 1.5 | 2 | 2.5 | 25.1 |
| 26 | 1.25 | 1.5 | 0.5 | 1.5 | 23.8 |
| 27 | 1.25 | 1.5 | 3.5 | 1.5 | 16.1 |
| 28 | 1.25 | 1.5 | 2 | 1.5 | 19.5 |

Consulting Set

| # | X1 | X2 | X3 | X4 | Y |
|---|---|---|---|---|---|
| 1 | 2 | 1.75 | 3 | 1.75 | 16.7 |
| 2 | 2 | 1 | 1.75 | 1 | 15.8 |
| 3 | 2 | 1 | 3.75 | 2.5 | 19.2 |
| 4 | 1.25 | 1 | 1.25 | 1.25 | 22.2 |
| 5 | 1.75 | 2 | 1.25 | 2 | 22.2 |
| 6 | 1.75 | 1 | 1.25 | 1 | 19.2 |
| 7 | 0.75 | 2 | 1.25 | 1 | 23.7 |
| 8 | 0.75 | 1 | 1.25 | 2 | 32.8 |
| 9 | 1.25 | 2.5 | 2 | 1.5 | 19.7 |
| 10 | 1.25 | 1.5 | 2 | 0.5 | 13 |




因为过度训练，consulting error出现轻微的恶化

Fig. 13. Overtraining of the FL net from chemical compounding data (real-world data).

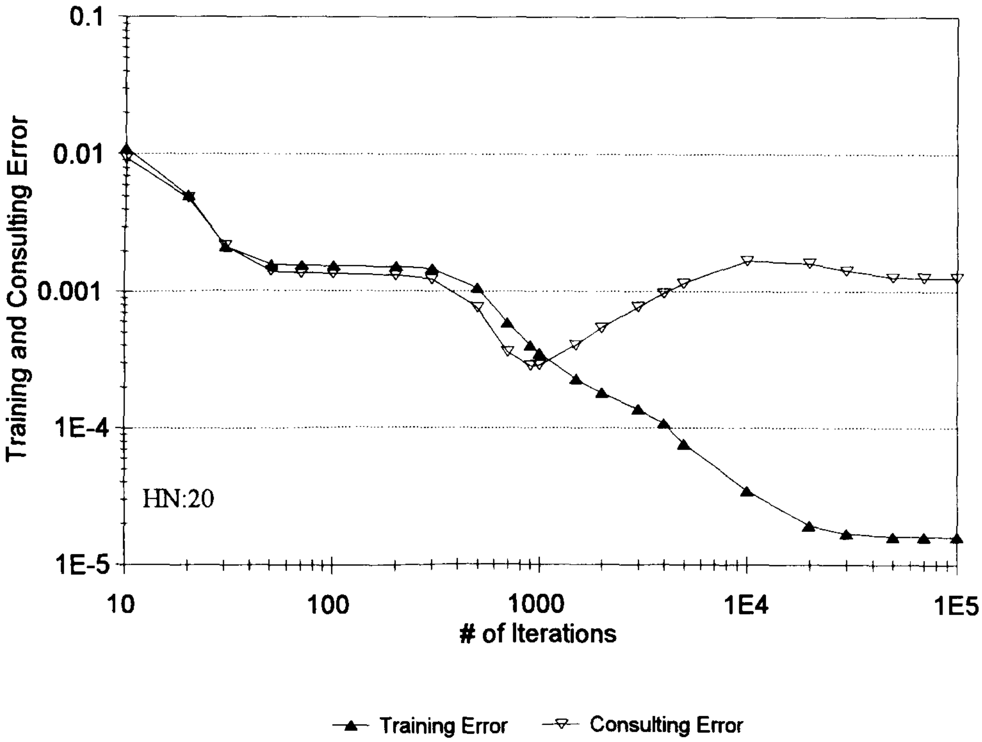



Fig. 14. Overtraining of GDR net from chemical compounding data (real-world data).

## 4. Concluding remarks

The objective of this paper is to promote increased understanding of some related matters in the general topic area of learning and generalization. Some of these are listed in the following:

(1) Adequacy of the feedforward net architecture does not imply that the GDR approach must be used in training. Other variations exist.

(2) Even in the absence of noise, estimation errors in generalization mode are generally not as small as those attained for the training set.

(3) In the absence of noise, generalization errors are due to inadequacy by interpolation or extrapolation and cannot be reduced by further training. They can be reduced by increase in the quality of the training set, such as making the exemplars more representative of the behavior of the function to be learned.

(4) For real-world data with usually a stochastic element in the function, overtraining can occur. Overtraining is aggravated through increase in the number of parameters made available or through use of excessive number of adaptive iterations in training. 增加参数和迭代次数都会造成过度训练引起过拟合

(5) The random-vector FL net trains simply and rapidly and is guaranteed to converge to the optimal solution in a known number of iterative steps.

## Postscript

Rigorous justification of the random-vector functional-link approach has since been obtained by B. Igelnik and Yoh-Han Pao. Part of that justification is presented in "Additional perspectives on feedforward neural-nets and the Functional-link", by B. Igelnik and Yoh-Han Pao, IJCNN'93, Nagoya, Japan (Oct. 1993) [14] with additional material in [15] and [16].

## Appendix 1. Observations on rapid convergence attainable with the random vector FL net

A typical RV Functional-link net is depicted in *Fig. 2(b)* of text. It is useful and important to note that the output of an augmentation node is a scalar, but that scalar is a function of the entire input pattern vector. In a function mapping $R^n \to R$, the output is $o = \sum \beta_j g(\boldsymbol{a}_j^t \boldsymbol{x} + b_j)$.

In practice, we have the original vector components as well as the augmentation nodes. In the interest of simplicity in representation we do not distinguish between the two different types of input nodes in the following formalism. That is, all the weights are denoted $\beta_j$, each one of which weights the output of the corresponding node $j$ and all of the weighted values are summed to yield the output. 为了表示简单，我们在下面的形式中不区分原始输入节点和增强节点这两种类型的输入节点

The scalar output $o$ is computed as $o = \boldsymbol{B}^t \boldsymbol{d}$, where

$$\boldsymbol{B} = [\beta_1\, \beta_2 \ldots \beta_{N+J}]^t\,, \quad \text{and}$$
$$\boldsymbol{d} = [\delta_1\, \delta_2 \ldots \delta_{N+J}]^t\,.$$

The symbol $J$ denotes the total number of augmentation nodes and $N$ denotes the number of components of the original input vector.

For the augmentation nodes, $\delta_m = g(\mathrm{net}_m)$ for $m = N+1, ..., N+J$ where $\mathrm{net}_m = a_{m1}x_1 + ... + a_{mN}x_N + b_m = \boldsymbol{a}_m^t \boldsymbol{x} + b_m$,

$$\boldsymbol{a}_m^t = [a_{m1}, a_{m2}, ..., a_{mN}]^t , \qquad \text{and}$$

$$\boldsymbol{x} = [x_1, x_2, x_3, ..., x_N]^T .$$

每一个增强节点都是由全体原始输入乘以权重向量再加一个偏置项组成，随后在经过一个函数。

The $a_m$, $b_m$ parameters are selected at random but are scaled to avoid saturation of $g(\cdot)$. For pattern $\boldsymbol{x}^{(p)}$ we get $\mathrm{net}_m^{(p)} = \boldsymbol{a}_m^t \boldsymbol{x}^{(p)} + b_m$, $\delta_m^{(p)} = g(\mathrm{net}_m^{(p)})$ and $o^{(p)} = \boldsymbol{B}^t \boldsymbol{d}^{(p)}$.

For a given set of input and target data pairs $\{\boldsymbol{x}^{(p)}, t^{(p)}\}$, $p = 1, 2, ..., P$, the task is to determine (learn) unknown $\beta$ parameters. The learning process can be posed as an optimization problem, where the criterion function $E$ defined as

$$E = \frac{1}{2P} \sum_{p=1}^{P} (t^{(p)} - \boldsymbol{B}^t \boldsymbol{d}^{(p)})^2$$

is to minimized.

输出t是一个标量，转置后不变，所以才会有如下等式

We observe that equations $t^{(1)} = \boldsymbol{B}^t \boldsymbol{d}^{(1)} = (\boldsymbol{d}^{(1)})^t \boldsymbol{B}$, $t^{(2)} = \boldsymbol{B}^t \boldsymbol{d}^{(2)} = (\boldsymbol{d}^{(2)})^t \boldsymbol{B}$, ..., $t^{(P)} = \boldsymbol{B}^t \boldsymbol{d}^{(P)} = (\boldsymbol{d}^{(P)})^t \boldsymbol{B}$ can be written in a compact form as $\boldsymbol{t} = \boldsymbol{V}\boldsymbol{B}$ where $\boldsymbol{t} = [t^{(1)}, t^{(2)}, ..., t^{(p)}]^t$ and $\boldsymbol{V} = [(\boldsymbol{d}^{(1)})^t, (\boldsymbol{d}^{(2)})^t, ..., (\boldsymbol{d}^{(P)})^t]^t$ so that criterion $E$ can be expressed as

$$E = \frac{1}{2P} (\boldsymbol{t} - \boldsymbol{V}\boldsymbol{B})^t (\boldsymbol{t} - \boldsymbol{V}\boldsymbol{B}) .$$

V = P * (N+J)是个矩阵
B = (N+J) * 1是个向量
t = P * 1是个向量
令y = t -VB
dE/dB = dE/dy * dy/dB，由dx^Tx/dx = 2x；dAx/dx = A^T
得dE/dB = -1/P * (t - VB) * V^T，其维度为 PX1 * (N+J)XP，发现维度不相容。这时体现了标量对多个向量链式求导法则的不同（y = t-VB,E = 1/(2P)y^Ty,B->y->E,E就是标量）此时dE/dB = (dy/dB)^T * dE/dy，详见刘建平博客园

The gradient is accordingly

$$\boldsymbol{r} = \frac{\partial E}{\partial \boldsymbol{B}} = -\frac{1}{P} \boldsymbol{V}^t (\boldsymbol{t} - \boldsymbol{V}\boldsymbol{B})$$

and the learning process can be written as

$$\boldsymbol{B}^{\text{new}} = \boldsymbol{B}^{\text{old}} + \frac{\eta}{P} \boldsymbol{V}^t (\boldsymbol{t} - \boldsymbol{V}\boldsymbol{B}^{\text{old}}) ,$$

这是针对一个输出的情况，当输出扩展到h维，t=P * h，t仍是以列向量的形式参与训练t(1),t(2)....t(h) = P *1维，每个维度分别得到最优的B(1),B(2),....B(h)

where $\eta$ is used to minimize $E$ in the $r$ direction.

Search for an optimal $\boldsymbol{B}^*$ in the negative direction of the gradient of $E$ is usually efficient for a quadratic $E$ function but there is some waste motion due to the zig-zagging nature of the search process.

The waste motion can be eliminated with use of the Conjugate Gradient (CG) method, a first-order indirect optimization process. That method uses gradient information for finding better directions for search. In all cases of quadratic optimization the system optimum can be reached in a *finite number of iterations* but the CG method guarantees convergence in a specific predetermined number of operations. 共轭梯度法见知乎收藏

According to the CG methodology, the parameter update is defined by

$$\boldsymbol{B}^{\text{new}} = \boldsymbol{B}^{\text{old}} + \eta \boldsymbol{s} .$$

Directions of search $\boldsymbol{s}$ are computed at every point $\boldsymbol{b}^{(\cdot)}$ as

$$\begin{array}{lll} \text{At} & \boldsymbol{B}^{(0)}: & \boldsymbol{s}^{(0)} = -\boldsymbol{r}^{(0)} \\ & \boldsymbol{B}^{(1)}: & \boldsymbol{s}^{(1)} = -\boldsymbol{r}^{(1)} + \gamma_1 \boldsymbol{s}^{(0)} \\ & \boldsymbol{B}^{(2)}: & \boldsymbol{s}^{(2)} = -\boldsymbol{r}^{(2)} + \gamma_2 \boldsymbol{s}^{(1)} \\ & \ldots & \ldots\ldots \\ & \boldsymbol{B}^{(N+J-1)}: & \boldsymbol{s}^{(N+J-1)} = -\boldsymbol{r}^{(N+J-1)} + \gamma_{N+J-1} \boldsymbol{s}^{(N+J-2)} \\ & \boldsymbol{B}^{(N+J)}: & \textit{True optimum} \text{ reached not later than this} \end{array}$$

where $M = N + J$ is the dimension of the parameter vector $\boldsymbol{B}$, and

$$\gamma_1 = \frac{\|\boldsymbol{r}^{(1)}\|^2}{\|\boldsymbol{r}^{(0)}\|^2}, \qquad \gamma_2 = \frac{\|\boldsymbol{r}^{(2)}\|^2}{\|\boldsymbol{r}^{(1)}\|^2},$$

and so on.

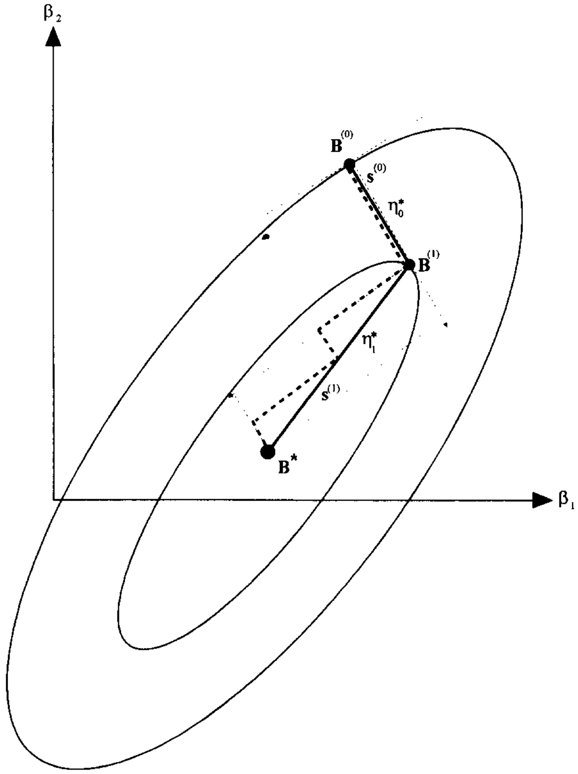



Fig. A.1. A comparison of two search strategies: --- quadratic descent minimization, —— conjugate gradient method, --- lines used to establish direction for conjugate gradient method.

In general,

$$\boldsymbol{B}^{(\lambda+1)} = \boldsymbol{B}^{(\lambda)} + \eta \boldsymbol{s}^{(\lambda)}$$

$$\boldsymbol{s}^{(\lambda)} = -\boldsymbol{r}^{(\lambda)} + \frac{\|\boldsymbol{r}^{(\lambda)}\|^2}{\|\boldsymbol{r}^{(\lambda-1)}\|^2} \boldsymbol{s}^{(\lambda-1)}$$

$$\text{for } \lambda = 0, \; s^{(0)} = -r^{(0)}$$

$$\text{and } \lambda = K, \; B^{(K)} \text{ is optimum}, \; K \leq N + J.$$

In *Fig. A.1*, we demonstrate graphically the difference which occurs during the minimization of a quadratic function of two variables ($M = 2$) using the directions of search

(a) along the negative gradient of the criterion function $E$ or
(b) along the direction computed according to the CG methodology.

## References


[1] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximations, *Neural Networks* 2 (1989) 359–366.

[2] Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Networks* 2 (1989) 183–192.

[3] P. Cardaliaguet and G. Euvrard, Approximation of a function and its derivative with neural network, *Neural Networks* 5 (2) (1992) 207–220.

[4] Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks* (Addison-Wesley, Reading, MA, 1989).

[5] M.S. Klassen, Y.H. Pao and V. Chen, Characteristics of the Functional link net: A higher order delta rule net, *Proc. IEEE Annual Internat. Conf. on Neural Networks*, San Diego, CA (1988).

[6] Y.H. Pao, Functional link nets: Removing hidden layers, *AI Expert* 4 (4) (1989) 60–68.

[7] Y.H. Pao and Y. Takefuji, Functional-link net computing: Theory, system architecture, and functionalities, Special issue of *Computer* on Computer Architectures for Intelligent Machines, IEEE Computer Society Press, Vol. 3 (1991) pp. 76–79, also published in Readings in *Computer Architectures for Intelligent Systems* (expanded version), J. Herath (ed.), in press.

[8] Y.H. Pao, S.M. Phillips and D.I. Sobajic, Neural-net computing and the intelligent control of systems, Special Intelligent Control issue of *Internat. J. Control* 56 (2) (1992) 263–289 (Taylor and Francis, London, UK).

[9] D.E. Rumelhart, G.E. Hinton and R.J. William, Learning internal representation by error propagation, in: D.E. Rumelhart and J.L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Foundations (MIT Press, Cambridge, NA, 1986) 318–362.

[10] D.S. Touretzky and G.E. Hinton, A distributed connectionest production system, CMU-CS-86-172, Carnegie-Mellon University Report, PA, 1986.

[11] R. Fletcher and C.M. Reeves, Function minimization by conjugate gradients, *Comput. J.* 7 (1964) 149–154.

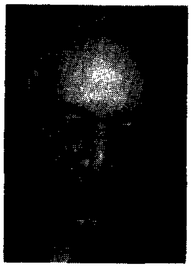
[12] M. Hestenes and E. Steifel, Methods of conjugate gradients for solving linear systems, National Bureau of Standard Report NO. 1659, 1952.

[13] C.L. Giles and T. Maxwell, Learning invariance and generalization in high-order neural networks, *Applied Optics* 26 (1987) 4972–4978.

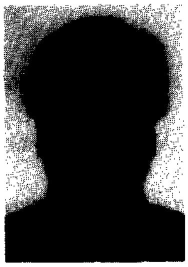
[14] B. Igelnik and Y.H. Pao, Additional perspectives of feedforward neural-nets and the functional-link, Technical Report 93-115, Center for Automation and Intelligent Systems, Case Western Reserve University, 1993, also in *Proc. IJCNN'93*, Nagoya, Japan (Oct. 25–29, 1993) 2284–2287.

[15] B. Igelnik and Y.H. Pao, Efficient learning in the Functional-Link net: Explicit construction of one layer feedforward neural net and estimates of accuracy of approximation, submitted to *IEEE Trans. Neural Networks*.

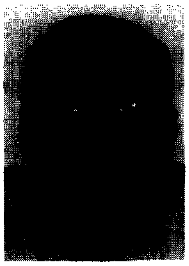
[16] B. Igelnik and Y.H. Pao, Random vector version of the Functional-Link net, submitted to the *28th Annual Conf. on Information Sciences and Systems*, Princeton University, Princeton, NJ (March, 1994).



**Yoh-Han Pao** has been a Professor of Electrical Engineering and Computer Science at Case Western Reserve University (CWRU) since 1967. He has served as chairman of the University's Electrical Engineering Department (1969–77), as Director of the Electrical, Computer and System Engineering Division at NSF (1978–1980), and as founding director of the Center for Automation and Intelligent Systems Research at CWRU. He is the George S. Dively Distinguished Professor of Engineering at CWRU, is a Fellow of IEEE and of the American Optical Society. He has been a NATO Senior Science Fellow; has visited MIT, Edinburgh University, and the Turing Institute as lecturer/researcher, and has been a member of the technical staff at AT&T Bell Laboratories in Murray Hill, New Jersey. He is co- founder and president of AI Ware Inc., Cleveland, Ohio.



**Gwang-Hoon Park** received the B.S. and M.S. degrees in the Dept. of Electronic Engineering from Yonsei University in Seoul, Korea in 1985 and 1987, respectively, and also M.S.E.E. degree in the Dept. of Electrical Engineering and Applied Physics from Case Western Reserve University in 1990. He is now Ph.D. candidate in the Dept. of Electrical Engineering and Applied Physics in CWRU. His current research interests are neural net computing, image and signal processing, adaptive control and evolutionary programming.



**D.J. Sobajic** (M'80–SM'89) received the B.S.E.E. and the M.S.E.E. degrees from the University of Belgrade in Yugoslavia and the Ph.D. degree from Case Western Reserve University, Cleveland, Ohio. At present, he is with the Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland. He is also Engineering Manager of AI WARE, Inc. His current research interests include power system operation and control, neural net systems and adaptive control. Dr. Sobajic is a member of the IEEE Task Force on Neural Network Applications in Power Systems and of the IEEE Intelligent Controls Committee. He is the Chairman of the International Neural Networks Society Special Interest Group on Power Engineering.