Analytica Chimica Acta, 247 (1991) 97–105
Elsevier Science Publishers B.V., Amsterdam

Comparison of the training of neural networks for quantitative x-ray fluorescence spectrometry by a genetic algorithm and backward error propagation

M. Bos * and H.T. Weber

University of Twente, Department of Chemical Technology, P.O. Box 217, 7500 AE Enschede (The Netherlands)

(Received 26th June 1990)

Abstract

Neural networks are shown to be useful as empirical mathematical models in the calculation of quantitative analytical results, giving sufficient accuracy to compete successfully with various common calibration procedures. The performance of these neural-network models for calibration data from x-ray fluorescence spectrometry (XRF) was evaluated for two training methods, i e , backward error propagation (BEP) and a genetic algorithm (GA) For a small training set (13 members) of data from Fe/Ni/Cr samples taken from the literature, the BEP-trained models compared favourably with other literature methods. The GA-trained models performed poorly for these samples. The two models performed equally well when trained on a larger data set (30 members) consisting of XRF data for thin Fe/Ni layers on a substrate, for which both the composition and the thickness were determined The predictive power of both models for samples outside the range of the training set was unsatisfactory

Keywords X-ray fluorescence spectrometry; Calibration, Neural networks

The main aim of this paper is to show that neural networks can be applied in quantitative chemical analysis as accurate calibration models for multivariate data analysis. Other aims are to establish the dimensions of the neural network representation that are needed to obtain the required accuracy and to investigate the differences between backward error propagation and genetic training of neural networks.

In a previous paper [1], it was shown that singular value decomposition and the Ho–Kashyap algorithm (SVDHK) can be used successfully in multivariate data analysis for quantitative x-ray fluorescence spectrometry. In this SVDHK method, a general model is used to take into account the interactions between the various constituents of the sample in terms of the cross-products of the fluorescence intensities at the lines of the elements [2]. For x-ray fluorescence, these interactions can be predicted and modelled on theoretical grounds. Neural networks are, in principle, capable of coping with such interactions automatically if they are trained with sufficient samples that exhibit these interactions [3].

X-ray fluorescence data from the literature [4] were chosen for this investigation because these allow the new techniques to be compared with well established procedures for which the performance has already been ascertained. But the neural network approach was also used for Fe/Ni thin-film samples to show that the method can replace more complicated theoretical models which can only be solved iteratively [5].

THEORY

Neural network theory and the backward error propagation rule have been fully explained [6]. An example of how to apply this theory to quantita-

0003-2670/91/$03 50

tive chemical analysis has been given [7]. Genetic algorithms constitute a set of general and efficient means of solving optimization problems in very large multidimensional search spaces [8–11]. These algorithms are adaptive generate-and-test procedures derived from the principles of natural population genetics [11]. General theory underlying these procedures has been reported [8–11]. In this section, the application of this theory to the design of a training procedure for neural networks is described.

The topography of the neural network used here is given in Fig. 1. The neurons are ordered in three layers and fully connected. All neurons in the first or input layer are connected to all external inputs. All neurons in the second or hidden layer have the same set of inputs, i.e., the set of outputs of the neurons in the input layer. The output of each neuron of the hidden layer goes to every neuron in the third or output layer which uses these signals as inputs. The outputs of this third layer constitute the set of external outputs.

Every neuron in the network is a signal-processing unit that calculates a value of its single output based on the set of values of its input signals and a set of internal parameters. The set of internal parameters consists of a set of weighting factors, a bias and a so-called temperature factor T. The set of weighting factors consists of one term for each input of the neuron. Mathematically, the behaviour of a neuron can be described by

$$\text{output} = 1/\left\{1 + \exp\left[-\left(\left(\sum_{j=1}^{n} \text{input}_j w_j\right)/T\right) + \text{bias}\right]\right\} \tag{1}$$

where w_j is the weighting factor of the jth input, T the temperature factor and n the number of inputs of the neuron. The action of a neuron is completely specified by the set of parameters, $w_1 \cdots w_j$, bias and T; from now on, this set is denoted by $\mathbf{x}$. The operation of a complete network of neurons can be calculated if its topography and all vectors $\mathbf{x}$ are known.

The procedure for calculating the external outputs of a network for a given set of external inputs is as follows: (1) calculate the outputs of the neurons of the input layer, using the external inputs; (2) calculate the outputs of the hidden layer, using the outputs of the first layer as inputs to these neurons; and (3) calculate the external outputs using the outputs of the hidden layer as inputs to the neurons of the output layer.

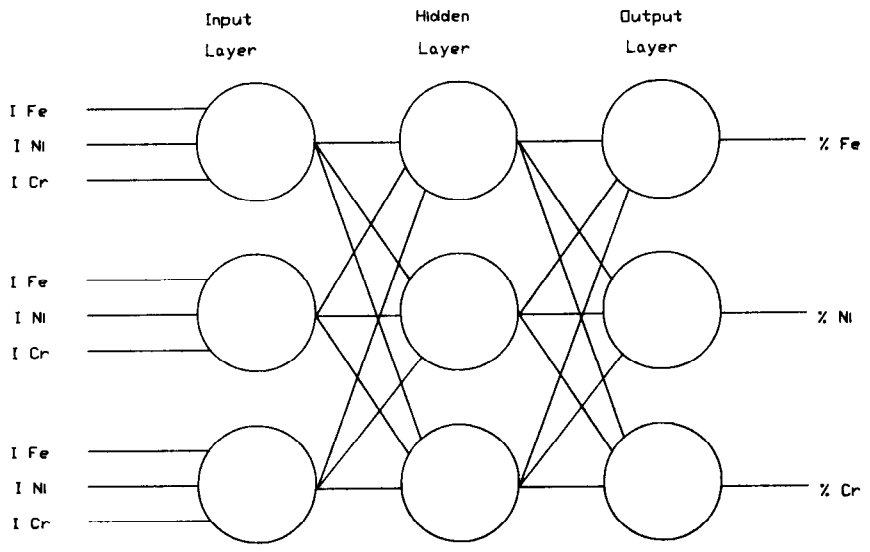

Fig 1 Topography of a fully connected forward-feed 3 × 3 × 3 neural network I is relative intensity.

Fitness

The fitness of a neural network is a measure of how well it produces the correct external outputs over a number of sets of external input values belonging to samples for which the real output values are known (the training set). Mathematically, this fitness is expressed by

$$\text{fitness} = 1/\left[\sum_{j=1}^{k}\sum_{i=1}^{l}(\text{required output} - \text{calc. output})^2\right] \quad (2)$$

The summation is over the l external outputs of the network ($i = 1 \cdots l$) and over the number of samples k ($j = 1 \cdots k$).

Training

Training of a neural network is the process in which the internal parameters of the neurons which form the network (the vectors **x**) are adjusted in order to maximize the fitness. A suitable algorithm is the well known rule for backward error propagation. This algorithm is known to produce suboptimal results in some cases in which the response surface of the fitness as a function of the internal parameters is multinodal. Here, this backward error propagation rule for training a neural network is compared with a training procedure based on a genetic algorithm.

The genetic algorithm is iterative and starts with a pool of neural networks for which the initial internal parameters are chosen at random. The fitness of each member of the pool for the training set is then calculated. The next step is analogous to the natural selection process. Based on their fitness values, pool members have a chance of reproduction that is given by the factor

$$f = \text{fitness/average fitness of whole pool} \quad (3)$$

Pool size is kept constant and the total number of offspring is limited to a certain fraction of the pool size. This mechanism ensures that the less successful networks will disappear from the pool. If no other genetic operators were applied, this process would fill the whole pool with copies of the most successful network that was present initially. However, two other genetic operators are applied after this reproduction step, i.e., crossover and mutation.

For the crossover mechanism, the neural networks in the pool are considered as a chromosome containing the numbers representing the internal parameters of each neuron placed in a long row. Crossover is done by exchanging the corresponding parts of two separate rows belonging to two pool members. The position in the row where the exchange takes place is chosen at random at bit level, to ensure efficient exchange of the properties between two neural networks in the pool. This mechanism allows an efficient search of a very large parameter space [11]. However, not all regions of the parameter space can be reached, because the space searched is limited to all combinations of the parameters that are present in the initial pool. To extend this search space, the mutation operator is needed. In the mutation mechanism, a parameter in the chromosome of a randomly chosen member of the pool is changed randomly at a random position on the chromosome.

After the mutation and crossover operations have been applied, a new iteration cycle is started in which the fitness of the new members of the pool is evaluated, the reproduction process is continued, etc. The genetic algorithm is stopped when the fitness of the best pool member exceeds a threshold value or when a maximum number of iterations is reached.

PROGRAM DEVELOPMENT

Neural network training by backward error propagation (BEP) and genetic algorithms (GA) and the required data structures for the neuron, the neural network layer, the network itself and the genetic pool are very suitable for implementation in an object-orientated computer language such as SMALLTALK. Moreover, this language offers rapid prototyping facilities for constructing a user-friendly interface with windows plus graphics. Therefore, the first version of the program was written in SMALLTALK/V running on an 8-MHz XT-clone with an 8087 coprocessor. The program

offers the following features: variable dimensioning of the neural network, BEP with adjustable learning rate, moment factor and temperature adjustment rate, variable dimensioning of the genetic pool, and GA with adjustable crossover and mutation rates. The program provides pulldown menus and produces numerical as well as graphic outputs for the error per sample (BEP) or the fitness of the genetic pool members (GA). Because the program was very slow, it was not suitable for the production runs needed for the investigation but it served well as a template for the second version of the program and as a debugging aid in this later development.

The genetic algorithms rely heavily on a random number generator; it is important that it should have a long cycle. The random number generator used here was as described earlier [12].

The production version of the program was written in C language and offers the same functions as the SMALLTALK program, but lacks the sophisticated user interface. There are no graphics and the user input is via a question/answer module. The program was tested to run on the Digital Equipment Microvax 3100 and on personal computers if compiled with the Microsoft C 5.1 compiler.

EXPERIMENTAL

The measurements of the Fe/Ni thin-film samples sputtered on a silicon substrate were made with a Philips x-ray fluorescence spectrometer (PW1480) against pure iron and pure nickel standards. A flow counter was used as detector. The x-ray tube had a chromium target and was operated at 50 kV. The dispersing crystal was LiF 200, the collimator setting was fine, and intensities were measured at the Fe K_α and Ni K_α lines (Fe $2\theta = 57.58$ and Ni $2\theta = 48.69$)

RESULTS AND DISCUSSION

The Fe / Ni / Cr system

The data used were the relative intensities and weight fractions as given by Rasberry and Heinrich [4], which are reproduced in Table 1. The relative intensities of the elements were autoscaled [13] over the training set to a zero mean an a standard deviation of 1.0 for each element separately before they were used as input values for the neural network. The weight fractions were also normalized before use in the trainingset, but to a mean value of 0.5 and a standard deviation of

TABLE 1

XRF data for the Fe/Ni/Cr system [a]

Sample	Relative intensity			Weight fraction		
	Fe	Ni	Cr	Fe	Ni	Cr
4184	0 3751	0 0002	0.4572	0.6322	0.0000	0.3658
4014	0.0013	0 4319	0.4392	0.0000	0.6064	0.3883
5074	0.4663	0.0216	0.3336	0.6838	0 0498	0 2525
5181	0.5123	0.0443	0 2721	0 6945	0 0996	0 1988
5324	0.3678	0.0933	0 3393	0 5280	0 1927	0.2696
5321	0.4512	0.0949	0 2662	0 5919	0 2002	0.1988
7271	0.5378	0.0362	0 2599	0 7159	0 0829	0.1879
161	0.1522	0.4535	0 2129	0 7159	0 0829	0.1879
1189	0 0135	0.5852	0 2348	0 0140	0 7260	0.2030
3987	0.4484	0.4287	0.0007	0 3431	0 6552	0.0000
5054	0.4844	0.0006	0.3422	0 7250	0 0015	0 2577
5202	0.4649	0.0684	0 2839	0.6303	0 1480	0 2130
5364	0.3313	0.1175	0 3450	0.4721	0 2357	0 2784
1188	0 0706	0.5689	0 1831	0.0660	0 7265	0.1540

[a] From Rasberry and Heinrich [4]

1/12. The performance of the various neural network topographics and training methods were established by the leave-one-out method, i.e., the complete data set is used for training, except the data on the sample for which the composition has to be predicted.

For the training by the genetic algorithm, a neural network was coded in a chromosome as one large array containing the weighting factors, temperatures and biases of all the neurons in sequence as 8-byte floating point variables of double type. The mutation operator was used in the form of a generator producing random numbers in the interval -10.0 to $+10.0$ of double type (8 bytes) at randomly chosen positions at the parameter level in the genetic pool.

For the crossover operation, two different random integers were generated with a value less than or equal to the number of chromosomes in the genetic pool. These two numbers determined which chromosomes were to be used in the crossover process. Then a third random integer number was generated to decide the point at bit level where the crossover was to take place. Finally, the two corresponding parts of the two chromosomes were exchanged.

Table 2 shows the results of the predictions of the compositions of the unknown samples by neural networks trained by the genetic algorithm (GA) and the backward error propagation rule (BEP) in comparison with the SVDHK method [1] and the Rasberry and Heinrich (R&H) calculations [4]. The BEP results were obtained with a network topography of three input neurons, three hidden neurons and three output neurons that were fully connected in a forward-feed manner. Larger networks were investigated for training with this method, but the results did not differ significantly, so the $3 \times 3 \times 3$ network was used in production runs because it took less time to train. Training parameters were: learning rate 0.9, momentum 0.4 and temperature factor adjustment rate 0.1. The training procedure was continued until the sum of squared errors for the concentrations of the three elements over all training samples was less than $1.0E-5$ for the normalized data. Two sets of GA

TABLE 2

Comparison of the composition of Fe/Ni/Cr samples predicted by various methods

Sample	Weight fraction of metal					
	Real [a]	R&H	SVDHK	BEP	GA $6 \times 6 \times 3$	GA $3 \times 3 \times 3$
Iron						
5054	0 7250	0 7229	0 7212	0.7224	0.6985	0.6625
5202	0 6303	0.6277	0.6328	0.6289	0.6248	0.6421
5364	0.4721	0.4750	0.4662	0 4750	0 4550	0.4262
1188	0 0660	0 0655	0.0618	0 0705	0.1022	0 0625
3987	0.3431	0 3378	–	0.3565	0.3259	0.0755
Nickel						
5054	0 0015	0 0015	0.0054	0 0162	0.0212	0 0695
5202	0.1480	0 1507	0 1456	0.1463	0 1444	0.1343
5364	0.2357	0.2479	0.2450	0 2373	0.1708	0.2374
1188	0 7265	0.7236	0.7421	0 7270	0.6798	0 7238
3987	0.6552	0.6512	–	0.5666	0 6248	0 7075
Chromium						
5054	0.2577	0.2522	0 2586	0.2541	0 2480	0 2594
5202	0.2130	0.2090	0 2023	0.2124	0 2166	0.2171
5364	0.2784	0 2675	0.2767	0 2799	0.2765	0 2679
1188	0.1540	0.1421	0.1434	0 1441	0.1683	0 1861
3987	0 0000	0.0000	–	0 0869	0.1677	0 1753

[a] See Table 1.

results were obtained, one with a network topography of six input neurons, six hidden neurons and three output neurons, and the other with a $3 \times 3 \times 3$ network. The larger network was investigated because training of the $3 \times 3 \times 3$ network by GA suffered from premature convergence. In the GA training, the pool size was 50. The other training parameters (mutation rate and crossover rate) were changed halfway through the training from 12 mutations and 75 crossovers per generation to 25 mutations and 150 crossovers. Training was continued until the best fitness in the genetic pool exceeded 1.E + 4 or did not change significantly ($< 5\%$) over 20 000 iterations (premature convergence).

It can be seen in Table 2 that the prediction of the composition of sample 3987 is poor whether the networks were trained by BEP or by GA. This happened because this is the only sample in the data set that contains no chromium; whereas all other samples contain $\geqslant 15\%$ chromium. Therefore, it can be concluded that the extrapolative power of the neural-network calibration models to values far away from the training set of data is poor for both methods. Sample 5054 also lies outside the range of the training set, but less so than sample 3987. The errors in the predictions for this sample are still unacceptable, but less erratic. This indicates a gradual degradation in the performance of the neural network outside the range of the training set.

A very important feature of the neural-network calibration model is that the predictions for samples outside the range of the training set show compositions outside this range, so that their unreliability is easily noticed.

TABLE 3

Comparison of discrepancies between the results from wet chemical analysis and the tested methods

Method	Average discrepancy [a]	Standard deviation [b]
R&H	-1.88×10^{-3}	6.39×10^{-3}
SVDHK	-5.92×10^{-4}	7.69×10^{-3}
BEP	4.92×10^{-4}	5.79×10^{-3}
GA $6 \times 6 \times 3$	-8.51×10^{-3}	2.79×10^{-2}
GA $3 \times 3 \times 3$	-1.62×10^{-3}	3.32×10^{-2}

[a] Average discrepancy between the chemically determined value and the predicted value [b] Standard deviation of the discrepancies.

Table 3 shows a comparison of the mean discrepancy and the standard deviation of the discrepancies between the predicted and chemically determined contents for all three elements (Fe, Ni and Cr) in samples 5054, 5202, 5364 and 1188. The BEP-trained neural network shows the smallest standard deviation and the smallest mean discrepancy. Although the differences from the R&H and SVDHK methods are statistically not significant, the neural network approach has the advantage that interactions of the measured signal do not have to be modelled explicitly, but are taken care of automatically.

The results obtained with neural networks trained by GA in Tables 2 and 3 force the conclusion that the GA method is inferior to all others for this data set.

Thin-film Fe / Ni samples

The complete data set used in the calculations for these samples is given in Table 4. This data set was split in two ways into a training set and a set of samples which were treated as unknowns. In the first split (set A), samples 1, 6, 9, 14, 21, 27 and 29 were chosen as unknowns; this represents a cut of the data lying well within the range of the other samples that were used for training. The second split (set B) was used to test the extrapolative power of the neural network model. Here, the unknowns were samples 32–36, which are clearly outside the range of the samples used for training (except for samples 12 and 13).

Table 5 shows the sum of the squared absolute prediction errors (SSQ) for various sizes of network if the training by BEP is continued to a sum of squared errors of less than 3.0×10^{-4} for the normalized training set data. Comparison of the sum of squared prediction errors shown for sets A and B proves that the extrapolative power of the BEP-trained neural networks is poor. The best topography for extrapolation is $3 \times 3 \times 3$.

Table 6 shows a comparison between the real composition and thickness and their predicted values for the samples that were treated as unknowns in set A, calculated with the BEP-trained $5 \times 5 \times 3$ network. Table 7 shows the predicted and real

TABLE 4

Relative intensities, composition and thickness of sputtered Fe/Ni film samples

Sample	Relative intensity		Content (%)		Thickness (Å)
	Fe K_α	Ni K_α	Fe	Ni	
1	0.00077	0 00380	14.8	85.2	210
2	0.00224	0 00924	17.1	82.9	530
3	0 00255	0 01175	15 6	84.4	660
4	0 00296	0.01545	14 0	86.0	860
5	0.0293	0.01200	17 2	82 8	690
6	0.00093	0 00380	17.4	82 6	220
7	0 00244	0 00971	17 6	82.4	560
8	0 00304	0.01199	17 8	82 2	700
9	0.00383	0.01468	18.1	81.9	860
10	0.00466	0 01728	18.6	81.4	1020
11	0.00514	0.01963	18.1	81.9	1150
12	0 00608	0.02310	18 1	81 9	1360
13	0 00892	0.03429	17 7	82 3	2040
14	0 00430	0.01708	17.5	82 5	1000
15	0.00426	0.01683	17 6	82 4	980
16	0.0455	0.01803	17.6	82 4	1050
17	0.00516	0 01751	19.9	80 1	1050
18	0.00301	0 01926	11.7	88 3	1040
19	0.00300	0.01932	11 6	88.4	1040
20	0.00528	0.01715	20.7	79.3	1040
21	0 00202	0.00928	15 7	84 3	520
22	0 00211	0.00921	16.4	83 6	520
23	0 00192	0.00894	15.5	84.5	500
24	0 00210	0.00919	16.4	83.6	520
25	0.00205	0 00933	15 8	84.2	530
26	0.00215	0 00931	16.5	83 5	530
27	0 00200	0 00909	15 9	84 1	510
28	0 00214	0.00883	17 1	82.9	510
29	0 00103	0.00367	19 2	80 8	220
30	0 00088	0 00367	17 1	82 9	210
31	0.00076	0 00387	14.5	85 5	210
32	0.00061	0.00404	11.5	88.5	210
33	0.00823	0 02544	21.3	78.7	1580
34	0.00717	0.02734	18.0	82.0	1620
35	0.00608	0.02799	15 4	84 6	1600
36	0.00540	0.02951	13 3	86 7	1640

data for set B treated as unknowns, calculated by the BEP-trained $3 \times 3 \times 3$ neural network. Except for sample 32, all samples have one or more predicted values outside the range of the training set, so that in practice there is some indication of their unreliability. In both tables, the sums of the iron and nickel contents are close to 100%, a feature that both neural networks seem to have learned from the training examples.

TABLE 5

Predicted results for BEP-trained neural networks of various sizes

Network	SSQ prediction	
	Set A ($\times 10^2$)	Set B ($\times 10^5$)
3×3×3	6.6	1 8
4×4×3	3.1	2 5
5×5×3	1 4	5.9
6×6×3	2 7	3.6
7×7×3	8.6	3 8
8×8×3	25.0	7 5
9×9×3	14.0	12 0

TABLE 6

Comparison between real data and data predicted by a BEP-trained $5 \times 5 \times 3$ neural network for set A

Sample	Fe (%)		Ni (%)		Thickness (Å)	
	Real	Pred	Real	Pred.	Real	Pred
1	14.8	14 97	85.2	85 02	210	205 4
6	17.4	17 31	82 6	82.69	220	218 7
9	18.1	18 09	81 9	81.89	860	856.7
14	17 5	17 42	82 5	82 55	1000	992.7
21	15 7	15.68	84.3	84 33	520	522.5
27	15.9	15.84	84.1	84 17	510	512.9
29	19.2	18.45	80 8	81 54	220	213.4

TABLE 7

Comparison between real data and data predicted by a BEP-trained $3 \times 3 \times 3$ neural network for set B

Sample	Fe (%)		Ni (%)		Thickness (Å)	
	Real	Pred.	Real	Pred.	Real	Pred
32	11.5	12 73	88 5	87 26	210	218 4
33	21 3	20 07	78.7	79 92	1580	1387 0
34	18 0	17 99	82 0	82.00	1620	1408 0
35	15 4	15 13	84 6	84.86	1600	1395 0
36	13 3	13 27	86 7	86.72	1640	1399.0

The results of prediction by two sizes of network trained by the genetic algorithm are given in Table 8. The results for set B show clearly that the extrapolative power of the $6 \times 6 \times 3$ network is extremely poor; the smaller network produces more realistic predictions. From this and similar findings with networks trained by BEP, it can be concluded that the smaller networks offer a pre-

TABLE 8

Comparison between real data and data predicted by the GA-trained 6 × 6 × 3 and 3 × 3 × 3 neural networks for sets A and B

Sample	Fe (%)			Ni (%)			Thickness (Å)		
	Real	6 × 6 × 3	3 × 3 × 3	Real	6 × 6 × 3	3 × 3 × 3	Real	6 × 6 × 3	3 × 3 × 3
Set A									
1	14.8	14 78	14.64	85.2	85 09	85.13	210	246.0	206.7
6	17.4	17.08	16.24	82.6	83.52	82.52	220	271.9	226.4
9	18.1	18 21	18.29	81.9	81 77	81.82	860	942.2	880.6
14	17.5	17 63	17.60	82.5	82 56	82.31	1000	1107.1	1000.7
21	15.7	15 72	15.47	84.3	84 37	84.37	520	443.7	508.0
27	15.9	15.90	15.63	84.1	84 20	84 16	510	439.5	502.6
29	19.2	18.80	17.41	80.8	82 36	80.61	220	290.5	252.7
Set B									
32	11 5	11.84	13.00	88 5	88 40	87 06	210	170.7	155.0
33	21 3	5.23	20.82	78.7	84 57	78.89	1580	2475.3	851 4
34	18 0	4.45	18.03	82.0	94 57	82.37	1620	2491.3	1168.5
35	15 4	15.79	4.37	84 6	95 49	84 52	1600	2493.5	1227.8
36	13 3	4.36	12.40	86 7	95 50	87.63	1640	2487.6	1098.4

dictive performance that deteriorates more gradually outside the range of the training set than is the case for the larger networks. The results of prediction by the 6 × 6 × 3 network for unknown samples inside the range of the training set (set A) are just as good as those obtained by by a BEP-trained network.

Comparison of the BEP rule and GA shows no particular advantage for the latter with regard to the extrapolative power of the networks. The BEP method is 3–5 times better in the efficiency of the training process. There is, however, a striking difference in the value of the squared errors summed over all training samples to which the models had to be trained by both methods to obtain similar results in prediction. Figure 2 shows this phenomenon for the prediction of the composition of sample 1188 (Table 1) by networks trained by both methods. the GA-trained networks seem to be more general representations of the calibration data than the BEP-trained networks. This difference can be explained by the fact that the GA method is driven by the performance of the model for the complete training set, whereas the model in the BEP method is adjusted for better performance for each member of the training set individually. Especially for large networks, this difference can lead to BEP-trained networks that closely model the noise in the data without capturing the underlying first principles.

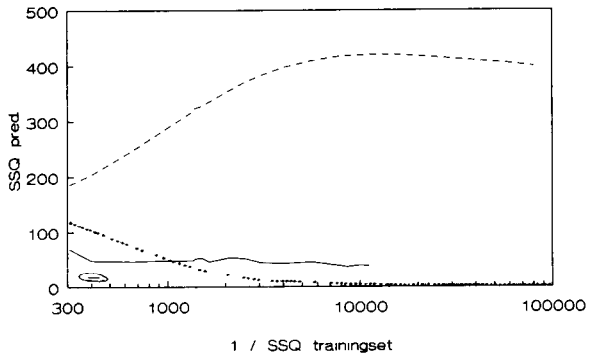

Fig. 2. Sum of squared prediction errors for Fe/Ni/Cr sample 1188 vs. best-fitness progress in training (- - - - - -) 6×6×3 network, BEP training, (·) 3×3×3 network, BEP training, (———) 6×6×3 network, GA training; the encircled line near the origin is for the 3×3×3 network with GA training.

REFERENCES

1 M. Bos, Anal Chim. Acta, 166 (1984) 261.
2 H J Lucas-Tooth and C. Pyne, Adv. X-Ray Anal , 7 (1964) 523
3 D E. Rumelhart and J.L McClelland, Parallel Distributed Processing, Vols. 1 and 2, MIT Press, Cambridge, MA, 1986.

4 S D Rasberry and K.F.J Heinrich, Anal Chem , 46 (1974) 81
5 J H.H G van Willigen, H T. Weber, M Bos and W.E van der Linden, Anal Chim. Acta, 136 (1982) 379
6 J L McClelland, D E. Rumelhart, Explorations in Parallel Distributed Processing, MIT Press, Cambridge, MA, 1988
7 M. Bos, A Bos and W E van der Linden, Anal Chim. Acta, 233 (1990) 31
8 J.H Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
9 K. de Jong, IEEE Trans. Syst Man Cybern, 10 (1980) 556
10 J.J Grefenstette, Machine Learning, 3 (1988) 225.
11 J.M Fitzpatrick and J.J. Grefenstette, Machine Learning, 3 (1988) 101.
12 B Wichmann and D. Hill, Byte, 12(3) (1987) 127.
13 K Varmuza, Pattern Recognition, Springer, Berlin, 1980, p 102