A Novel Algorithm for Local Minimum Escape in Back-Propagation Neural Networks: Application to the Interpretation of Matrix Isolation Infrared Spectra

Christoph Klawun and Charles L. Wilkins*

Department of Chemistry, University of California, Riverside, Riverside, California 92521-0403

Received March 1, 1994[⊗]

Back-propagation training of feed-forward neural networks often results in convergence to local minima, especially when multioutput networks and large training sets are employed. These local minima manifest themselves in a small number of residual output errors >95% (opposite errors). A procedure called "Flashcard Algorithm" has been developed in this research to overcome the opposite errors through overrepresentation of the difficult examples. A new and more objective criterion also defines a sufficiently low output error to avoid unnecessary training extensions. The problem of too slow convergence or untrainable network conditions has been addressed as well. It is now possible to predict a "doomed" run within the first few epochs in order to correct the number of hidden nodes and other network parameters. Applying the flashcard algorithm to train a neural network with matrix isolation infrared spectra resulted in 100% successful prediction of bond types and functional groups from training sets of 194 and 609 spectra, respectively. In addition, training times were cut drastically and reproducibility improved by more than an order of magnitude.

1. INTRODUCTION

The renaissance of neural networks in the later 1980s, fueled by new algorithms and adequate computer power, led many scientists to search for new applications and to refine existing network architectures and training algorithms. In the field of chemistry, the back-propagation (BP) training procedure by Rumelhart *et al.*[1,2] has gained wide popularity in spectral interpretation, despite its numerous shortcomings. Its applications include the prediction of electrophilic aromatic substitution reactions,[3] usage as a compact storage and retrieval medium for protein sequences,[4] and the interpretation of mass spectra.[5-8] Excellent reviews[9,10] and a book[11] cover the area of neural networks for chemists in more detail. This article focuses on a novel training procedure for multioutput feed-forward neural networks trained with the back-propagation algorithm and its application to the interpretation of matrix isolation infrared spectra.

Classification of Infrared Spectra. Use of artificial neural networks for the interpretation of IR spectra has many appealing aspects. The data range falls within a limited wavenumber window, and correlations between functional groups and band positions follow hundreds of rules, dependent on neighboring molecular features and environmental conditions; in short, this is a seemingly ideal case for nonlinear curve fitting algorithms. Since 1990, a number of papers have dealt with the classification of IR spectra using BP feed-forward neural networks[12-14] or self-organizing Kohonen maps.[8,15] The impact of different network architectures, spectral resolution, and overtraining effects on prediction quality has been studied,[16] and attempts were made to reconstruct IR spectra for selected functional groups from the network weights.[17,18] Neural networks have been implemented to serve as a retrieval system for 1129 IR spectra.[19] Small modular networks dedicated to interpreting particular substructures were reported to perform better than a huge flat architechture.[20,21] However, all research on IR spectral interpretation with neural networks has focused on readily available gas-phase or condensed-phase spectra. On the other hand, matrix isolation infrared (MI-IR) spectra feature much sharper bands due to drastically reduced rotational components and intermolecular interactions. Therefore, their classification with neural networks should produce more reliable results than those based on gas- or condensed-phase IR spectra. With the design of a gas chromatographic interface[22,23] in 1979, it became more feasible to collect MI-IR spectra, especially after the introduction of a commercial instrument[24,25] for analysis of complex mixtures. In this work, MI-IR spectra were subjected to neural network training and interpretation for the first time.

Neural Network Training Optimization. Since its introduction in 1986, the back-propagation (BP) training algorithm, which will be used throughout this paper, has been criticized for its slow convergence speed. In addition, it is prone to getting trapped in local minima. Much effort has been spent to remedy these shortcomings while retaining the simplicity and reliability of BP for those cases when it succeeds. Unfortunately, several papers concentrate on network architectures with only a single output node.[26-31] Although they report training acceleration by a factor of 2-100, this was achieved with a rather small number of input and hidden nodes. Improvements were made with a modified random optimization method[26] or by use of the conjugate gradient algorithm.[27,32,33] In some cases, routines controlling momentum and learning rate provided higher convergence rates,[30,32,34] but only a few authors optimized their BP results[32,33] for a fair performance comparison. Genetic algorithms in conjunction with BP training produced promising results, replacing the least important neuron weights with new random weights[35] or eliminating them altogether.[36] The opposite approach of "growing" nodes and hidden layers when needed also has been investigated.[37] However, it is much more important for a neural network to learn the examples correctly than to reach a low error margin in a shorter time. During a training session, the network usually falls into local minima on the error surface. A good training algorithm must therefore provide a means of escape from local minima[35,36,38-40] in order to increase the probability of finding the global minimum. The transition probability between a local and a global minimum also has been studied in depth.[41] With an increasing number of training examples and overdetermined problems, it seems that local

[⊗] Abstract published in *Advance ACS Abstracts,* June 1, 1994.

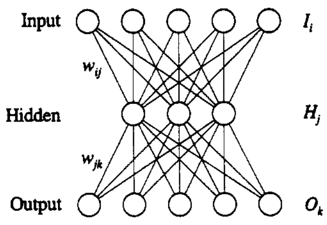


Figure 1. Feed-forward neural network architecture.

minima are inevitable. One paper[33] reports the training of 1700 phenomena on a network with 16 input, 20 hidden, and 7 output nodes. Only 92% of all examples could be learned, using a conjugate gradient algorithm. The BP algorithm leveled off after a mere 82% of the examples were trained satisfactorily. This work will show that it is possible to achieve 100% training even with a large number of output nodes and a considerable number of examples.

2. NEURAL NETWORK ARCHITECTURE

According to recommendations from previous work,[16] a feed-forward neural network with one input, one hidden, and one output layer was chosen, implemented in Borland Pascal 7.0 and trained using the "generalized delta rule" by Rumelhart *et al.*[1,2] (Figure 1). Its details are reproduced below. The input layer consisted of 752 input nodes I_i whose output values are normalized infrared absorption data between 680 and 4000 cm^{-1} (eq 1). Resolution varied between 1 cm^{-1} at 680 and 15 cm^{-1} at 4000 cm^{-1} (see section 3).

$$I_i^{\text{out}} = \text{Data}_i \tag{1}$$

Normalization was carried out by setting the largest absorption value to 1.0 and scaling the rest of the spectral data accordingly. Any negative values, *e.g.*, noise spikes, were set to zero prior to training. The size of the hidden layer H_j varied considerably. The classification of 8 bond types (section 4) required at least 6 units, and up to 50 hidden nodes were studied. For the training and interpretation of 35 different functional groups (section 5), the number of hidden nodes ranged between 18 and 80. The propagation of the input values through the network follows eqs 2–5

$$H_j^{\text{in}} = B + \sum_i (I_i^{\text{out}} w_{ij}) \tag{2}$$

$$H_j^{\text{out}} = \text{Sig}(H_j^{\text{in}}) \tag{3}$$

$$O_k^{\text{in}} = B + \sum_j (H_j^{\text{out}} w_{jk}) \tag{4}$$

$$O_k^{\text{out}} = \text{Sig}(O_k^{\text{in}}) \tag{5}$$

For a given hidden node H_j, all output values of the input layer I^{out} are multiplied by their connection weight w_{ij} (initially set to small random values) between H_j and I_i, added to a "bias node" B (=1.0), thus forming the input value H^{in} of the hidden node. Because H^{in} can assume rather large positive or negative values, it is scaled by a transfer function to yield a real number between 0.0 and 1.0. A sigmoidal transfer function (eq 6) also provides a discrimination effect whose

$$\text{Sig}(x) = \frac{1}{1 + e^{-\beta(x+\theta)}} \tag{6}$$

magnitude can be adjusted by varying β, the steepness of Sig(x) around $x = 0$. It is also possible to shift Sig(x) along the x-axis if $\neq 0$, but in this work Θ was always set to zero. The output nodes O_k are treated the same way to obtain output values between 0.0 and 1.0.

An O^{out} number close to 1.0 suggests a present functionality, and $O^{\text{out}} \approx 0$ implies an absent one. During the training process, the output of O_k is compared with the desired target value T_k for this node in order to compute an error signal O_k^{err} (eq 7).

$$O_k^{\text{err}} = O_k^{\text{out}}(1 - O_k^{\text{out}})(T_k - O_k^{\text{out}}) \tag{7}$$

$$\Delta w_{jk}(t) = \eta O_k^{\text{err}} H_j^{\text{out}} + \alpha \Delta w_{jk}(t-1) \tag{8}$$

$$w_{jk}(t) = w_{jk}(t-1) + \Delta w_{jk}(t) \tag{9}$$

Now the connection weights w_{jk} can be corrected to move O_k^{out} closer to T_k (eqs 8 and 9). How much of that error at current training step t will be incorporated in the weight change $\Delta w_{jk}(t)$ is determined by the learning rate η. In addition, the previous weight change $\Delta w_{jk}(t-1)$ is taken into consideration with a momentum term α. This circumvents training oscillation and helps in escaping from local minima early in the training session. Weights between the hidden and input layers w_{ij} are updated in a similar manner (eq 11 and 12), but H_j^{err} cannot be calculated in a straightforward manner, because no target values exist for H^{out}. Instead, eq 10 describes how to obtain error signals for hidden layer nodes.

$$H_j^{\text{err}} = H_j^{\text{out}}(1 - H_j^{\text{out}}) \sum_k (O_k^{\text{out}} w_{jk}) \tag{10}$$

$$\Delta w_{ij}(t) = \eta H_j^{\text{err}} I_i^{\text{out}} + \alpha \Delta w_{ij}(t-1) \tag{11}$$

$$w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t) \tag{12}$$

The root mean square (RMS) error over all output nodes for a training example (eq 13) provides a measure of how well this particular example has been learned by the network. After all examples have been presented to the neural network, the epoch error ϵ is the average of all RMS errors (eq 14). If ϵ reaches a small enough predetermined value or only drops by minuscule amounts, the network training is usually completed.

$$\text{RMS} = \left[\frac{\sum_{k=1}^{n}(T_k - O_k^{\text{out}})^2}{n}\right]^{1/2} \tag{13}$$

$$\epsilon = \frac{\sum_{x=1}^{m} \text{RMS}_x}{m} \tag{14}$$

Although it is appealing to minimize ϵ, this bears the danger of overtraining; *i.e.*, the network loses some of its generalization capabilities. As we will see in section 6, it is more desirable to set an upper limit to $|T_k - O_k^{\text{out}}|$ for all examples in order to determine a well trained neural network.

3. ADAPTING THE MATRIX ISOLATION IR DATABASE

Molecular Structures. The control software for Mattson Cryolect 4800 includes a database containing 5000 MI-IR spectra, including Wiswesser line notation[42] (WLN) structures, names, and CAS numbers for all compounds associated with the spectra. Because WLN structures do not lend themselves well to neural network training, an interpretation program was written in Borland Pascal 7.0 to convert them into a molecular graph representation. During this process, numerous discrepancies surfaced. Illegal structures were found, and molecular formulas generated from WLN did not correspond to the ones from the Cryolect database. All but five entries could be corrected.

Matrix Isolation FT-IR Spectra. The encrypted MI-IR spectra were decoded into 3680 16-bit data points per spectrum, using an algorithm obtained from Mattson Instruments. This translates into a wavenumber range from 451.48 to 3999.33 cm^{-1} and one data point every 0.964 cm^{-1}. However, all spectra were found to be normalized to 62 258 as the largest positive and −3277 as the largest negative absorbance value. In order to lose as little information as possible in any later data manipulation, the spectra were reduced back to their original scale with the data point differences found in the scaled spectrum. In addition, the region between 451 and 640 cm^{-1} contained only zeros in every spectrum; thus every spectrum was truncated to 3484 data points, starting at 640.49 cm^{-1}. In 5% of the spectra, significant base-line shifts from zero absorbance made a base-line correction necessary, and 24 other spectra of extremely poor quality had to be deleted altogether.

Another possible source of error, mislabeled spectra, had to be minimized before any neural network training could begin. For this purpose, we classified all 4916 usable structures into 64 common substructures and compared every spectrum against all other spectra in the database, compiling hit lists of 10 spectra in each case. If the substructures from the hit list did not correspond well with the ones from the reference spectrum, but correlated well among themselves, the reference spectrum was excluded from the neural network training pool. Although this procedure inherently introduced some subjectivity with the definition of substructure criteria, no unbiased comparison was possible due to the lack of other MI-IR spectra collections. After several iterations with rigorous conditions, the list of usable MI-IR spectra was reduced to 2825 entries.

Data Reduction. Obviously, 3484 data points or input nodes is overkill for neural network training. Although some researchers used fixed large wavenumber[14,17,18] or micrometer[14] increments, most authors[16,20,21] divide the cm^{-1} regime into a lower and higher resolution range or work with variable resolution.[12] An approach similar to the latter seems to be the most appealing, because it conserves needed resolution in the low-wavenumber range, while gradually condensing the higher wavenumber regions. Starting at $\nu_1 = 680$ cm^{-1} (eq 15), nonlinearity is achieved by decrementing in the $cm^{1/2}$ or $\mu m^{1/2}$ domain with $\tau = 3 \times 10^{-5}$ $cm^{1/2}$ (eq 16), which corresponds to $\tau \approx 1$ cm^{-1} at 680 cm^{-1}.

$$\mu_n = 1/\nu_n^{1/2} \quad (15)$$

$$\mu_{n+1} = \mu_n - \tau \quad (16)$$

$$\nu_{n+1} = 1/(\mu_{n+1})^2 \quad (17)$$

This result in a data reduction to 752 data points. Figure 2 illustrates the relation between resolution and wavenumber. The absorbance value for ν_{n+1} is calculated by averaging all wavenumber domain absorbance between ν_n and ν_{n+1}, excluding ν_n.

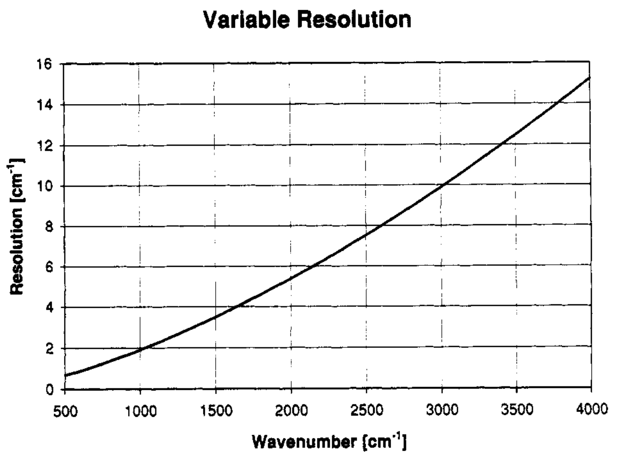


Figure 2. Data reduction with variable resolution: Correlation between wavenumber and resolution.

Table 1. Bond Types and Abundances in Training and Test Set

bond type	training set	test set
aliphatic ring	46	343
C–C	188	1091
C–H	192	1098
C–O	137	770
O–H	51	292
C=C	66	410
C=O	125	664
C≡C	23	0
total spectra	194	1100

4. BOND TYPE CLASSIFICATION

Selection of Bond Types. For preliminary testing of neural network training with MI-IR spectra, we chose a simple molecular representation, bond types, with the abundant atoms C, H, and O. This gave us some insight into the inner workings of back-propagation neural networks and their training behavior with MI-IR spectra. Out of 75 different possible bonds in the database, the bonds in Table 1 represent a large variety of molecules in 1294 different spectra. The separation into 194 training and 1100 test spectra was made in order to have a large enough test set for reliable statistics, and a training set small enough, so that it was feasible to examine every spectrum manually.

Training with Regular Back-Propagation. An initial training run with 50 hidden nodes, a learning rate of $\eta = 0.03$, a discrimination of $\beta = 1.0$, and a varying momentum α between 0.80 and 0.95 revealed a small number of *opposite* bond type predictions, *i.e.*, a prediction of >95% confidence for a present bond type, although in reality it was absent, and vice versa. Four reasons could be identified for such behavior: (1) over-/underrepresentation in training set, *e.g.*, 192 out of 194 training spectra contain C–H bonds, so it is hard for the neural network to classify the remaining 2 correctly (both H_2O); (2) local minimum in training reached; (3) molecule exhibits tautomerism or rearrangement; (4) wrong spectrum, or labile molecule destroyed in GC separation process.

Manual inspection of the corresponding spectra uncovered several bands violating IR selection rules or keto/enol tautomerism; those spectra were subsequently excluded from training. However, the errors >95% ("opposite errors") never disappeared completely, even after significant changes in learning rate and discrimination. A lower global epoch error level ϵ led to longer training times, but very few opposite errors disappeared.

Attempts to Improve Training. Adding slowly decreasing noise to the training spectra in order to avoid initial local

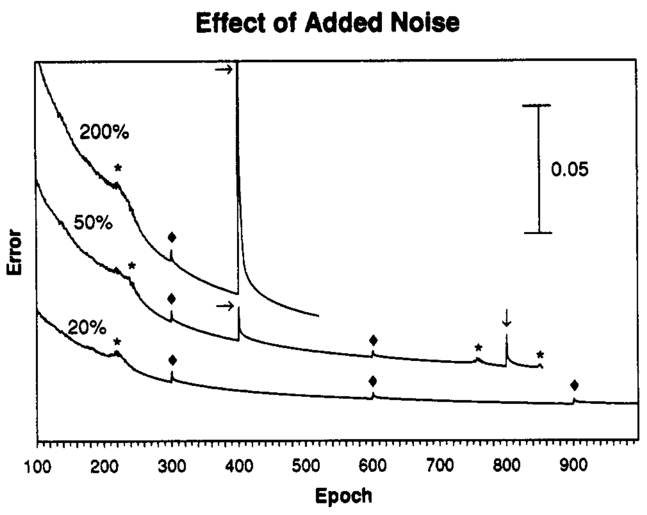


Figure 3. Effects of added noise ($\alpha = 0.90$, $\eta = 0.02$, $\beta = 1.0$, 50 hidden nodes). All three curves start at $\epsilon = 0.052$ and end at $\epsilon = 0.014$. The diamonds (♦) indicate a ±20% noise addition to all neural network weights, and the arrows (→) represent a ±50% and ±200% noise addition. Spontaneous conversion of a "opposite error" to a low error is marked with an asterisk (*).

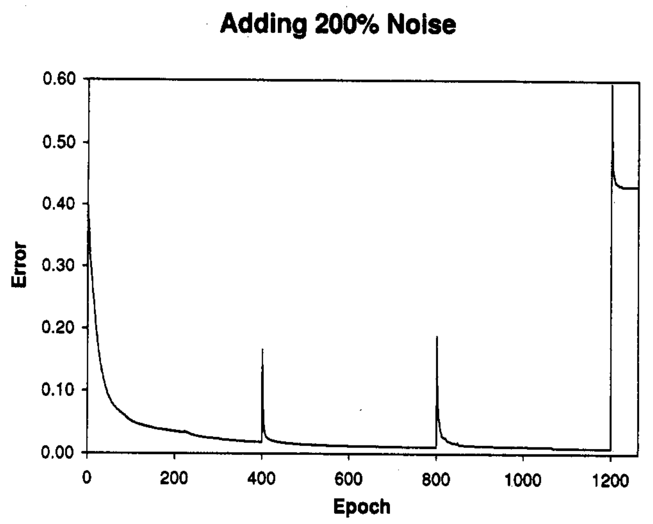


Figure 4. Disturbing neural network weights every 400 epochs with ±200% noise addition. After the third addition at 1200 epochs, the network has completely destabilized (for training parameters see Figure 3).

Table 2. Training Epochs and "Opposite Errors" for $\epsilon = 0.014$ ($\alpha = 0.90$, $\eta = 0.02$, $\beta = 1.0$, 50 Hidden Nodes)

same example sequence		random example sequence		double learning[a]	
epochs	opp. error[b]	epochs	opp error[b]	epochs	opp error[b]
972	4	658	3	1372	5
972	4	913	4	993	5
972	4	912	4	1355	6
972	4	904	4	993	5
972	4	737	3	585	4
972		825 ± 120		1060 ● 324	

[a] $\eta = 0.04$ between hidden and output node. [b] Opposite errors, predictions with >95% error.

Table 3. Training Improvement with the Flashcard Algorithm[a]

algorithm	epochs	improvement
regular back-propagation	824.8 ± 119.6 (±14.5%)	
flashcard algorithm	355.2 ± 7.56 (±2.1%)	56.9%
flashcard with extra training[b]	315.7 ● 4.06 (±1.3%)	61.7%

[a] The number of epochs reflects the additional training effort from overrepresenting selected spectra. [b] After all opposite errors have been reduced, overrepresent all spectra according to their largest output error (see text).

minima only prolonged the training, but did not produce the desired effect. We also attempted to disturb the network weights with varying noise levels, once the epoch error decline became minimal.[3,43] This helped to speed up the convergence of the *good* responses but did not convert any opposite errors. Figure 3 illustrates the network behavior when ±20, ±50, and ±200% noise is added to every neural network weight. Smaller disturbances are not very beneficial, while a large disturbance leads to much quicker convergence. An interesting phenomenon is the little noisy humps, marked with asterisks. A closer observation of all output errors for all 194 examples reveals that they can be attributed to a conversion of an opposite error; *i.e.*, one of the output errors >95% suddenly drops to values <30% within a few epochs. From the 200% curve in Figure 3, one could draw the conclusion that regular network disturbances with large noise levels will eventually bring the number of opposite errors to zero. However, this did not turn out to be the case. Pursuing the network training with ±200% noise addition every 400 epochs resulted in lower epoch errors (Figure 4), but, after the third disturbance at 1200 epochs, the network "blew up". Evidently, noise addition does not change the fundamental training process. It may be compared with a longer wrench to further tighten a bolt (the network): Too much leverage will eventually break it.

Random presentation of all examples during an epoch is important because it gives the network a better chance to find the global minimum.[31] Table 2 shows the epochs required to reach a final epoch error of $\epsilon = 0.014$. The initial weight matrix was set randomly to ±0.15 once and used as a starting point for all training runs. Convergence improvement was also attempted by doubling the learning rate between the hidden and the output layer,[44] but this produced the opposite effect.

Overrepresentation of "Difficult" Examples. While watching the number of opposite errors (>95%) and "large errors" (30–95%) after every epoch for all examples, an interesting observation could be made. The epoch error convergence leveled out as soon as all large errors had disappeared, leaving only a small persistent residue of opposite errors, which by that time were all >99.99%. Lifting the neural network out of that local minimum evidently required more specific work on these opposite errors to at least turn them into ~95% errors. Therefore the following algorithm was devised:

(1) When the number of errors between 30 and 95% drops below a small predefined threshold, train the neural network *only* with those examples causing errors >95%.

(2) Continue to present the opposite error examples to the neural network, until at least one of them drops to below 95%. Then, resume the regular training epoch.

All opposite errors treated this way are absorbed by the neural network and drop to small values within a few epochs. We call this the "Flashcard Algorithm" in analogy to learning vocabulary words with flashcards: The difficult words have to be recalled more often, before they are memorized properly. The flashcard algorithm not only teaches the neural network every example it has been trained with but also reduces the required training time by 57% (see Table 3). In addition, the variance between training runs is lowered by an order of magnitude, making the back-propagation training much more reproducible. Figure 5 shows the epoch error progression for both regular back-propagation and improved convergences with the flashcard algorithm. After approximately 70 epochs,

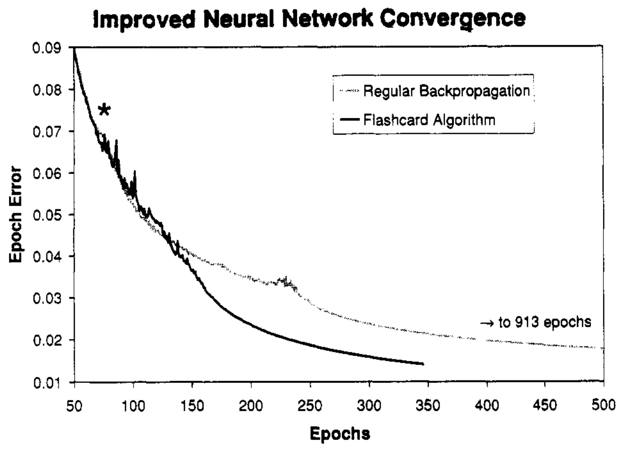


Figure 5. Comparison between regular back-propagation and flashcard algorithm training (for training parameters see Figure 3), showing improved neural network convergence. The astrisk (*) marks the point where the flashcard algorithm takes effect. In this particular example, the training time was cut from 14:06 to 5:24 h (on an Intel 486DX2-66 processor DOS protected mode).

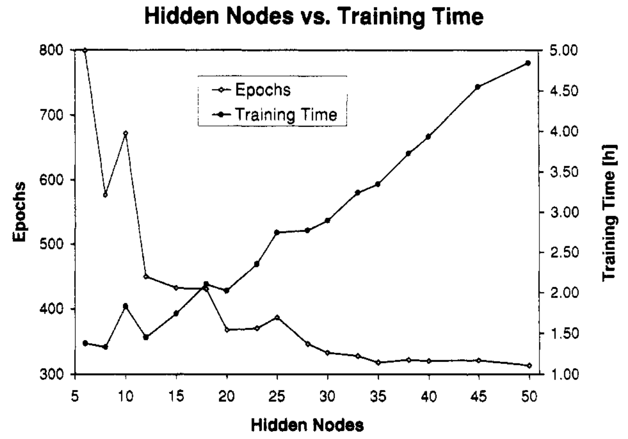


Figure 6. Effect of hidden nodes on epochs and training time (for training parameters see Figure 3). It was impossible to train the network properly with less than 6 hidden nodes.

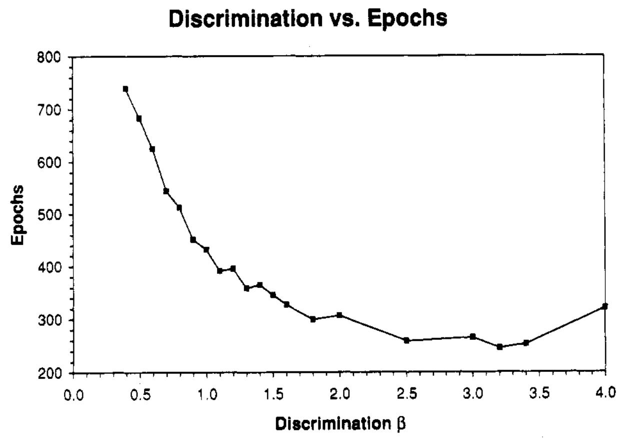


Figure 7. Relation between required training epochs and the discrimination β of the sigmoidal function (15 hidden nodes, α = 0.90, η = 0.02, ϵ = 0.014). Beyond β = 3.2 it was difficult to train the network to convergence, and past β = 4.0, several opposite errors could not be reduced.

the number of "large" output errors between 30 and 95% falls below a threshold of 5 and flashcard training begins. This threshold was selected by monitoring the number of large errors, which reached a plateau in the neighborhood of 5 during a regular training run. After all opposite errors have been eliminated, the overrepresentation of "difficult" spectra may be taken to its extremes, until a satisfactory epoch error ϵ has been reached. For this purpose, the largest of the eight output errors E_{max} for every spectrum determines the "degree of difficulty". Rounding $n = E_{max} \times 100 + 1$ to the nearest integer yields the number of times n which the spectrum has to be consecutively presented to the neural network. At the end of an epoch, E_{max} is reevaluated for all spectra. This results in an additional 4.8% decrease in training time to reach the same epoch error even faster (see Table 3). An approach similar to the flashcard algorithm has been reported in the literature,[31] where examples are removed from the training set, if they induce only minimal weight changes. However, that work deals with just two binary inputs and a single output node.

Optimization. The effects of changing the learning rate η and the momentum α on neural network training time and learning capabilities have been described in detail in the literature. Too large a learning rate or momentum will lead to oscillation and training divergence, and too small values result in slow convergence. Their actual values depend on the application, and we have found $\eta \approx 0.02$ and $\alpha \approx 0.90$ to produce satisfactory training. The number of hidden nodes relates to convergence speed, noise inertness, and neural network capacity, *i.e.*, how many different patterns a network can "remember". Figure 6 illustrates how too many hidden nodes slow down the training process without reducing the number of required epochs. On the lower end, the number of epochs suddenly shoots up, the training time levels out, and, with too few hidden nodes, the network cannot be properly trained at all. In other words, with 5 or less hidden nodes there was always one opposite error which could not be reduced, no matter how many times the according spectrum was presented to the network. With these observations, 15 hidden nodes are a good choice. The magnitude of random initial network weights does not influence the training progress at all. Any number between ±0.000 01 and ±10 resulted in approximately the same training time, although with initial weights of ±100 the number of epochs roughly doubled. Thus, we used ±0.1 for all subsequent training runs. No studies have been reported so far about the influence of β, the steepness of the sigmoidal function around its inflexion point, and many researchers just disregard it. As Figure 7 shows, the number of epochs decreases with higher β, but only to the point where the network cannot represent a sharper discrimination, as is the case with too few hidden nodes. A value of β = 2.5 was found to be satisfactory for this application to provide high discrimination and short network training. For the test set, higher values for β produced less "indicisive" outputs around 0.5, but the overall results for correct and false predictions remained the same.

Evaluation. Any pattern classifier evaluation entails the difficult task of *objectively* characterizing the method performance. To a large extent, this depends on the test set composition. Obviously, it is impossible to compile large training and test sets with even distribution and combination of all individual classifiers. As a consequence, a high population of, *e.g.*, C=C bonds in the test set will give a high *a priori* percentage of correct positive C=C classifications. For example, if 500 out of 1000 test spectra contain a C=C bond and the pattern classifier positively identifies a C=C bond in 50% of all cases, this is equivalent to the outcome of random guessing and no additional information has been gained. On the basis of Rotter and Varmuza's information gain calculations,[45] Wilkins *et al.*[46] propose a figure of merit M, which does not depend on the test set composition and gauges the pattern classifier ability to distinguish between a present and an absent feature. M is calculated using the *a*

Table 4. Bond Type Classification with 1100 Test Set Spectra

		prediction (%)				
		positive		negative		
bond type	present in *m* spectra	correct	a priori[a]	correct	a priori[a]	figure of merit
aliphatic ring	343	49.3	31.2	92.2	68.8	0.171
C—C	1091	99.9	99.2	22.2	0.8	0.152
C—H	1098	100.0	99.8	100.0	0.2	1.000
C—O	770	94.6	70.0	90.0	30.0	0.609
O—H	292	69.2	26.6	90.8	73.5	0.301
C═C	410	65.1	37.3	84.6	62.7	0.197
C═O	664	96.1	60.4	96.3	39.6	0.764
C≡C	0			98.7	100.0	
overall	1100	89.8	53.0	92.5	47.0	

[a] A priori probability of correct "guessing", *i.e.*, bond type abundance in the test set.

Table 5. Misclassification Errors per Test Set Spectrum

errors	spectra	percentage
0	570	51.8
1	323	29.4
2	165	15.0
3	36	3.3
4	6	0.5
total	1100	100.0

priori uncertainty or entropy $H(A)$ which depends on the test set composition, and the *a posteriori* entropy $H(A|B)$, the residual prediction uncertainty after applying the pattern classifier. The difference between them is the information gain $I(A,B)$ (eq 18).

$$I(A,B) = H(A) - H(A|B) \tag{18}$$

When the pattern classifier produces the same uncertainty $H(A|B)$ as random guessing $H(A)$, the information gain $I(A,B)$ is obviously zero. However, even for the perfect classifier with no residual *a posteriori* uncertainty, the information gain depends upon the test set composition $H(A)$. Dividing $I(A,B)$ by $H(A)$ in eq 19 removes this dependency and yields M, the figure of merit.

$$M = I(A,B)/H(A) \tag{19}$$

Table 4 summarizes the test set results with conventional correct present and absent predictions and also reports M for all bond type discriminations except C≡C (for a broader discussion of M and calculation of $H(A)$ and $H(A|B)$, see ref 46). Any neural network outputs <0.5 were taken as "absent" and otherwise as "present" predictions. It is not too surprising that the neural network has some difficulty in classifying the aliphatic ring correctly. Cyclopropanes are lumped together with cyclohexanes, cyclodecanes, and multiring structures, certainly a diffuse picture. The same is true for the mediocre O—H prediction, because it encompasses alcohols as well as carboxylic acids. On the other hand, the neural network easily recognizes the prominent C═O bands, and for the lopsided C—H bond distribution it produces a perfect score. The distribution of classification errors in Table 5 shows that only a small percentage of the test spectra have been grossly misclassified with three or more out of eight possible errors. These results underscore the importance of good feature selection prior to any neural network training. Therefore, in the next section, a wider variety of functional groups with more training spectra not only refines the neural network training but also probes the robustness of the flashcard algorithm.

5. FUNCTIONAL GROUP CLASSIFICATION

Selection of Functional Groups. Initially, the set of 2825 usable MI-IR spectra was categorized into 64 different substructures and functional groups. In order to minimize the detrimental effect of underrepresented features in neural network training, the following restrictions were applied to yield a selection of 35 functional groups (Table 6): (1) only molecules with the atoms C, H, N, and O; (2) no atomic bonds with less than 10 spectral examples; (3) no functional groups with less than 20 spectral examples.

Enhanced Local Minima Prevention. Initial attempts to train the neural network with the flashcard algorithm described in section 4 were disappointing. No combination of learning rate, number of hidden nodes, or discrimination steepness succeeded in reducing the number of opposite errors to zero. It seemed that the network was too "stupid" to handle the increased load of spectral examples and classifier features. A closer examination of the tough residue of opposite errors revealed that all of them ranged between 99.5 and 100.0%. Observing the conversion of opposite errors during a training run, we noted that the "easy" opposite errors around 95% were reduced relatively quickly, while the "hard" ones around 99% were driven even closer to 100%. Once they reached ~99.98%, they were impossible to turn around. The fact that those "extreme errors" only appeared after a few epochs led to the following enhancement of the flashcard algorithm:

(1) After every epoch, check for extreme output node errors >99.5%.

(2) Present the spectra associated with those errors to the neural network, until *all* of them exhibit errors <99.5%.

(3) Resume regular training.

With this cap on extreme errors, the neural network was able to learn all 609 training spectra correctly (see Figures 8 and 9). The simple flashcard algorithm took effect after the number of errors between 30 and 95% dropped below 100, and the set of corresponding spectra were presented to the network, until at least two of the opposite errors >95% disappeared. The interpretation of the results obtained from a test set of 1328 spectra (Table 7) follows along the same lines as those in section 4 for the bond type training. The training and test set spectra for cyclopropane and double bonds without any hydrogen atoms show either weak or no characteristic bands at all, in addition to their low abundance in the training set, which explains their low interpretation values. Although C≡N groups give rise to a very characteristic band around 2250 cm^{-1}, their poor recognition performance again can be attributed to their low abundance in the training set—clearly a drawback of feed-forward neural networks. On the other hand, the rare exocyclic C═C double bonds fare much better, presumably because they always fit one of the other C═C categories too. Comparison of the higher positive prediction success of the "generic OH" category with the OH bond type prediction (Table 4) supports this assumption: A positive OH prediction is usually accompanied with a positive identification of alcohols, phenols, or carboxylic acids, while the "stand alone" C≡N bond cannot be supported by anything else. A different molecule representation[8] or modular networks[8,20,21] will probably give better results. Optimizing these predictions will be addressed in future work, because it was not within the scope of this paper.

6. TRAINING OPTIMIZATION

Application to Large Training Sets. For very large training sets, it may not be practical to keep track of large (30–95%),

Table 6. Functional Group Types and Abundances in Training and Test Set

functional group	training set	test set	functional group	training set	test set
CH_3—C	408	1031	endocyclic C═C	33	61
C—CH_2—C	348	893	C≡N	13	6
C_3CH	122	317	ketone	63	157
C_4C	63	78	aldehyde	60	59
cyclopropane	11	11	carboxylic acid	61	64
cyclopentane	55	67	ester	130	479
cyclohexane	87	159	amide	55	49
C—CH_2—OH	61	115	C—NH_2	59	102
C_2CH—OH	62	73	C_2NH	34	39
C_3C—OH	18	17	C_3N	47	47
phenol	60	56	aromatic 6-ring	261	407
ether	94	206	aromatic 6-ring monosubstituted	68	190
RHC═CH_2	64	72	aromatic 6-ring *ortho*-disubstituted	56	48
RHC═CHR	42	155	aromatic 6-ring *meta*-disubstituted	22	22
R_2C═CH_2	25	27	aromatic 6-ring *para*-disubstituted	60	55
R_2C═CHR	46	100	pyridine ring	23	18
R_2C═CR_2	11	11	OH, generic	240	308
exocyclic C═C	15	8			
			total spectra	609	1328

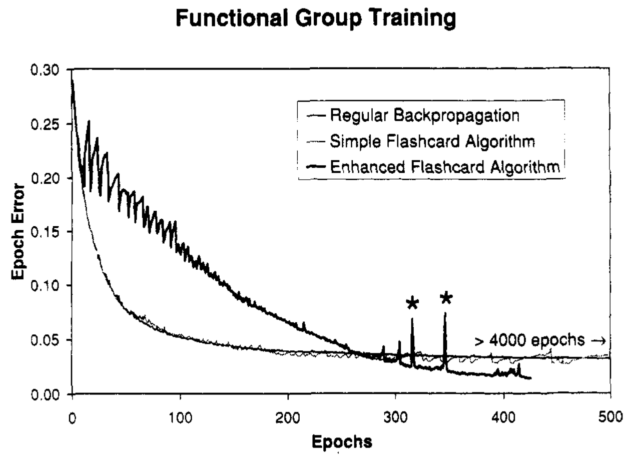
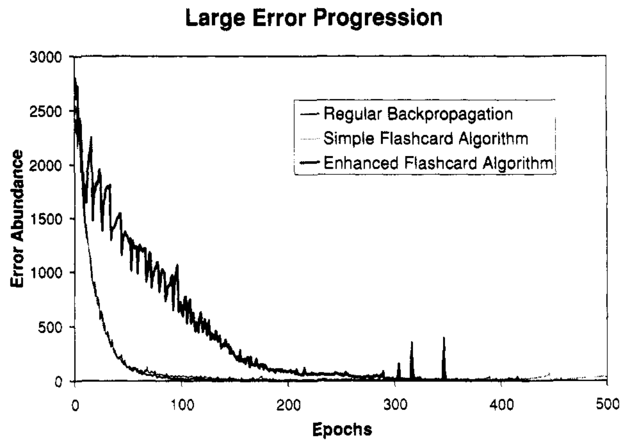


Figure 8. Comparison between regular back-propagation and the simple and the enhanced flashcard algorithm with extreme error correction (25 hidden nodes, $\alpha = 0.90$, $\eta = 0.04$, $\epsilon = 0.014$, $\beta = 1.0$), showing functional group training. The number of epochs for the flashcard runs are corrected for the additional training effort during overrepresentation. The large spikes marked with an asterisk (*) are due to an escape from a local minimum.



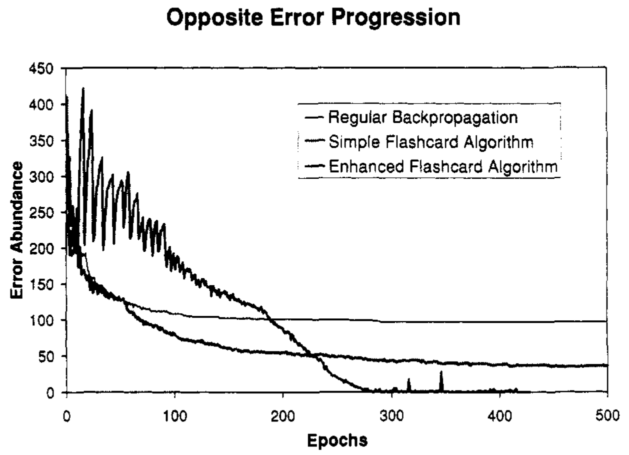


Figure 9. Development of errors during training runs as described in Figure 8: (a, top) Large errors between 30% and 95%; (b, bottom) opposite errors >95%.

opposite (>95%), and extreme output errors (>99.5%). Using the same set of 609 training spectra, we tested the feasibility of dealing with every occurring error >99.5% immediately, instead of after a completed epoch, and found only an insignificant increase in required epochs (406 ± 55 instead of 377 ± 60 epochs). However, too high a training rate causes the network to get trapped in a local minimum. For $\eta = 0.08$ (25 hidden nodes), the first error of 100.0% appeared after 8 epochs. While it could eventually be reduced to <99.5%, it caused too many 100.0% errors in the following epoch, and the network training stalled. Excessive overtraining of one single spectrum (several thousand consecutive presentations) seems to destabilize the network and send it into too deep a local minimum. But, with proper parameters, it should be possible to apply the flashcard algorithm to much larger training sets than the ones used in this paper.

Criterion for Completed Training. For any neural network applications, it is desirable to pursue the training process only as long as necessary. An overtrained network will recognize any training set spectrum with ease, but it loses its generalization capabilities for unknown spectra. One could run a test set through the network after every completed epoch[16] to catch the generalization apex, but this approach would require long training times. In addition, the network weights have to be saved to a hard disk every epoch in order to conserve all training states and later select the best. Therefore, many researchers just monitor the global epoch error ϵ to the point when progress has slowed down enough to stop the training run. Thus, a more objective approach is proposed, especially for multioutput neural networks: Neural network training is completed, when *all* output errors for *all* training examples have dropped below the "large error" limit of 30%. Clearly, this limit should be comfortably lower than 50%, but dropping it to values much lower than 10% does not improve anything. We observed that when there are only a few large errors left, they drop from 50 to 10% within a few epochs. Extending the

Table 7. Functional Group Classification with 1328 Test Set Spectra

functional group	present in *m* spectra	prediction (%) positive correct	positive a priori	negative correct	negative a priori	figure of merit
$CH_3—C$	1031	88.5	77.6	67.0	22.4	0.250
$C—CH_2—C$	893	78.6	67.2	67.4	32.8	0.158
C_3CH	317	60.3	23.9	85.9	76.1	0.170
C_4C	78	57.7	5.9	91.4	94.1	0.182
cyclopropane	11	9.1	0.8	98.6	99.2	0.016
cyclopentane	67	41.8	5.1	95.2	95.0	0.138
cyclohexane	159	53.5	12.0	88.2	88.0	0.137
$C—CH_2—OH$	115	63.5	8.7	98.4	91.3	0.408
$C_2CH—OH$	73	57.5	5.5	94.7	94.5	0.232
$C_3C—OH$	17	35.3	1.3	97.2	98.7	0.113
phenol	56	67.9	4.2	98.9	95.8	0.462
ether	206	61.2	15.5	96.0	84.5	0.320
$RHC{=}CH_2$	72	65.3	5.4	96.2	94.6	0.328
$RHC{=}CHR$	155	40.7	11.7	97.9	88.3	0.205
$R_2C{=}CH_2$	27	51.9	2.0	98.0	98.0	0.253
$R_2C{=}CHR$	100	33.0	7.5	91.0	92.5	0.056
$R_2C{=}CR_2$	11	18.2	0.8	98.9	99.2	0.060
exocyclic $C{=}C$	8	62.5	0.6	97.4	99.4	0.261
endocyclic $C{=}C$	61	39.3	4.6	95.8	95.4	0.133
$C{\equiv}N$	6	0.0	0.5	98.3	99.6	0.003
ketone	157	54.1	11.8	96.7	88.2	0.275
aldehyde	59	59.3	4.4	97.2	95.6	0.304
carboxylic acid	64	76.6	4.8	98.7	95.2	0.541
ester	479	87.5	36.1	97.3	63.9	0.643
amide	49	65.3	3.7	98.8	96.3	0.424
$C—NH_2$	102	68.6	7.7	97.6	92.3	0.421
C_2NH	39	66.7	2.9	96.7	97.1	0.329
C_3N	47	57.5	3.5	96.8	96.5	0.270
aromatic 6-ring	407	87.7	30.7	91.6	69.4	0.517
aromatic monosubstituted	190	71.6	14.3	98.1	85.7	0.482
aromatic *ortho*-disubstituted	48	64.6	3.6	97.5	96.4	0.352
aromatic *meta*-disubstituted	22	36.4	1.7	97.6	98.3	0.132
aromatic *para*-disubstituted	55	72.7	4.1	96.5	95.9	0.389
pyridine ring	18	27.8	1.4	98.5	98.6	0.104
OH, generic	308	87.0	23.2	87.4	76.8	0.423
overall	1328	73.1	11.8	95.5	88.2	

neural network training past this point for over 30 epochs lowered the global epoch error ϵ by a factor of 2 but did not result in any prediction improvements.

Noise Addition. Some researchers[3,43] have suggested that periodical noise addition to a neural network training process improves the probability of a local minimum escape and stabilizes its prediction performance. Constant noise addition during network training has been reported as well[47] but resulted in degraded performance with increasing noise levels. In order to probe the performance of the flashcard algorithm in the presence of noise, we added between ±0.04 and ±10% peak-to-peak noise to the spectral data during the training session (25 hidden nodes, $\eta = 0.03$, $\alpha = 0.9$, $\beta = 1.0$). No change in prediction capabilities could be noticed, and only for high noise levels, training times increased (see Table 8). Figure 10 shows how longer training times also leads to much lower final epoch errors, before the training runs at high noise levels could be completed.

"Doomed" Training Sessions. To date, the determination of a suitable number of hidden nodes remains tedious and vague, and the same is true for many other back-propagation neural network training parameters. Often, the training parameters are too strict or too relaxed. The latter guarantees incredibly sluggish training convergence, and the former leads to local minima, oscillations, and training "blow ups". Both cases slow down development and waste computer resources. It would be helpful to predict from the first few epochs, whether a training session is "doomed", abort it and start another session with adjusted parameters. We found that the number of extreme errors (>99.5%) provides a reliable gauge. If there are as many extreme errors as training spectra after the first epoch, the training parameters should not be any stricter (*e.g.*, high enough training rate η or discrimination factor β). On the other hand, if it takes too many epochs for the first extreme error to appear, the training will take longer than necessary, and some parameters should be tightened. Table 9 supports these observations with a variety of examples. If there are not enough hidden nodes, the number of extreme errors multiplies rapidly and the training gets stuck in a local minimum, endlessly presenting the same spectrum to the network without reducing a 100.0% error. Increasing the number of hidden nodes provides the neural network with more "capacity" for high values of η or β, and even with a very high initial number of extreme errors, training can still be completed.

Table 8. Effects of Constant Added Noise Levels during Training (25 Hidden Nodes, $\eta = 0.03$, $\alpha = 0.9$, $\beta = 1.0$)

noise level (±%)	final epoch error, ϵ	epochs	correct prediction (%) positive	correct prediction (%) negative
0.0	0.0202	386.1	73.2	95.6
0.04	0.0197	338.4	74.3	95.1
0.1	0.0194	339.2	74.2	95.3
0.4	0.0184	398.6	73.3	95.4
1.0	0.0183	336.2	73.6	95.7
4.0	0.0134	541.2	74.5	95.3
7.0	0.0095	2051.1	71.5	95.3
10.0		(2446)[a]		

[a] No convergence due to a large number of 100.0% errors, training aborted after 2446 epochs.

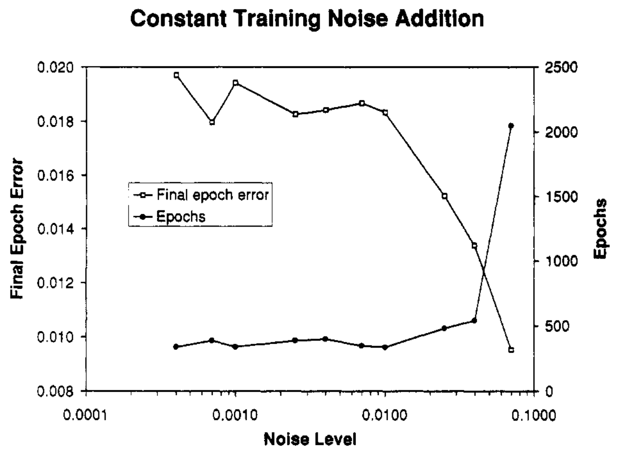


Figure 10. Effects of constant added noise levels during training (25 hidden nodes, $\eta = 0.03$, $\alpha = 0.9$, $\beta = 1.0$) on the final epoch error and required epochs.

7. CONCLUSIONS

In this paper, we successfully demonstrate the neural network training with matrix isolation IR spectra for the first time and find that it is difficult for a back-propagation multioutput network to learn every single spectrum of the training set successfully. The network always gets stuck in a local minimum, which manifest itself with a residual small number of large output errors >95% (opposite errors). These errors hamper the success rate by predicting with high confidence a present feature when it is actually absent and vice versa. Through overrepresentation of those "difficult" examples, the opposite errors could be eliminated, and the neural network was capable of correctly classifying every spectrum it had been trained with. This algorithm has been named flashcard algorithm in analogy of learning vocabulary words with flashcards: Hard words come up more often, until they are memorized satisfactorily. Compared to the regular optimized back-propagation algorithm, the flashcard algorithm also cuts the training time to reach the same final epoch error ϵ by more than 60% and improves the run-to-run training reproducibility by more than an order of magnitude to ±1.3%. When we increased the number of training spectra from 194 to 609, and the output nodes from 8 bond types to 35 functional groups, it became necessary to cap off extreme errors >99.5% during the training, before they turned into 100.0% errors, which are nearly impossible to convert. For this purpose, all spectra associated with extreme errors were presented either after epoch or immediately upon extreme error detection, until all errors were reduced to <99.5% and regular training could be resumed.

Adding constant low levels of noise to the training spectra during a flashcard training did not influence the neural network's ability to escape from a local minimum, nor did it enhance any prediction capabilities for the trained network. The magnitude of initial random weights was found to be insignificant; no change in training behavior could be observed for initial random weights between ±0.000 01 and ±10.

We replaced the common criterion for satisfactory trained neural networks—minimizing the global epoch error, until it shows only very little progress—with a more objective procedure: When all training set output errors have dropped to values <10–30%, the network training is considered complete, because there are no more local minima left to escape from. Extending the training beyond that point did not result in improved test set predictions.

Finally, a way of predicting doomed training sessions after a few epochs was devised. This common phenomenon occurs when the training parameters are not right. With too relaxed values, *i.e.*, a low learning rate η, momentum α, discrimination factor β, or too many hidden nodes, the training progress can be excruciatingly slow, while under too tight conditions neural networks often cannot be trained. The abundance and progression of extreme errors (>99.5%) reveals whether a run should be aborted and restarted using easier or tighter parameters, thus effectively eliminating some of the guesswork involved in choosing proper neural network training conditions.

Table 9. Effect of Various Parameters on Neural Network Training Success

training rate η	discrimination factor β	hidden nodes	extreme errors	after n epochs	total epochs	final epoch error ϵ
0.01	1.0	80	1	14	389.8	0.0209
0.03	1.0	80	228	1	262.0	0.0114
0.05	1.0	80	2411	1	240.7	0.0124
0.02	1.0	60	2	1	241.9	0.0179
0.02	1.5	60	507	1	195.9	0.0186
0.02	1.7	60	1449	1	242.2	0.0168
0.01	1.5	60	57	1	318.5	0.0219
0.01	2.0	60	726	1	299.8	0.0199
0.01	2.5	60	1195	1	271.9	0.0202
0.005	2.0	60	1	11	563.8	0.0215
0.005	2.5	60	258	1	475.7	0.0238
0.005	3.0	60	696	1	493.0	0.0237
0.02	1.0	40	2	7	265.8	0.0216
0.04	1.0	40	27	1	240.1	0.0133
0.06	1.0	40	104	1	198.9	0.0125
0.08	1.0	40	148	1	349.0[a]	0.0067
0.10	1.0	40	191	1	195.1	0.0119
0.01	1.0	25	1	17	688.2	0.0271
0.04	1.0	25	9	3	311.3	0.0163
0.07	1.0	25	13	1	337.9	0.0125
0.08	1.0	25	29	1	(6)[b]	
0.01	1.0	20	1	18	104.3	0.0293
0.03	1.0	20	1	5	606.6	0.0204
0.05	1.0	20	1	2	(19)[b]	
0.01	1.0	18	1	21	1426.1	0.0254
0.03	1.0	18	1	5	(26)[b]	
0.01	1.0	17	1	12	(42)[b]	

[a] Outlier due to an unfavorable set of initial weights and training spectra sequence. [b] Stuck with a large number of 100.0% errors at that epoch.

Together with the flashcard algorithm, this provides a means of faster and more reliable multioutput neural network training. Local minima are avoided, and it is possible without overtraining to build a neural network which can recall everything it has seen before.

ACKNOWLEDGMENT

This work has been carried out with support from ICI plc, United Kingdom, and the National Science Foundation, grant CHE-92-01277. The authors would like to thank Prof. Sharon L. Neal for access to seven 486DX2-66 computers, which greatly accelerated our progress. We also appreciate the generosity of ATI/Mattson for providing us with the decryption code for the Cryolect database. Part of this work was presented at the Twentieth Annual Meeting of the Federation of Analytical Chemistry and Spectroscopy Societies (FACSS XX) in Detroit, MI (October 1993), paper 476.

REFERENCES AND NOTES

(1) Rumelhart, D. E.; McClelland, J. L.; The PDP Research Group. *Parallel Distributed Processing—Explorations in the Microstructure of Cognition: Foundations*; MIT Press: Cambridge, MA, 1986; Chapter 8, pp 318–362.

(2) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536.

(3) Elrod, D. W.; Maggiora, G. M.; Trenary, R. G. Applications of neural networks in chemistry. 1. Prediction of electrophilic aromatic substitution reactions. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 477–484.

(4) Wu, C. H. Classification neural networks for rapid sequence annotation and automated database organization. *Comput. Chem.* **1993**, *17*, 219–227.

(5) Curry, B.; Rumelhart, D. E. MSnet: A neural network which classifies mass spectra. *Tetrahedron Comput. Methodol.* **1990**, *3*, 213–237.

(6) Lohninger, H. Classification of mass spectral data using neural networks. In *Software development in chemistry 5—Proceedings of the workshop "Computers in Chemistry", Oldenburg*; Gmehling, J., Ed.; Springer-Verlag: Heidelberg, 1991; pp 159–168.

(7) Lohninger, H.; Stancl, F. Comparing the performance of neural networks to well-established methods of multivariate data analysis: the classification of mass spectral data. *Fresenius J. Anal. Chem.* **1992**, *344*, 186–189.

(8) Gasteiger, J.; Li, X.; Simon, V.; Novič, M.; Zupan, J. Neural nets for mass and vibrational spectra. *J. Mol. Struct.* **1993**, *292*, 141–160.

(9) Zupan, J.; Gasteiger, J. Neural networks: A new method for solving chemical problems or just a passing phase? *Anal. Chim. Acta* **1991**, *248*, 1–30.

(10) Gasteiger, J.; Zupan, J. Neural Networks in Chemistry. *Angew. Chem., Int. Ed. Engl.* **1993**, *32*, 503–527.

(11) Zupan, J.; Gasteiger, J. *Neural Networks for Chemists—An Introduction*; VCH, Weinheim, 1993.

(12) Robb, E. W.; Munk, M. E. A neural network approach to infrared spectrum interpretation. *Mikrochim. Acta [Wien]* **1990**, *1*, 131–155.

(13) Munk, M. E.; Madison, M. S.; Robb, E. W. Neural network models for infrared spectrum interpretation. *Mikrochim. Acta [Wien]* **1991**, *2*, 505–514.

(14) Fessenden, R. J.; Györgyi, L. Identifying functional groups in IR spectra using an artificial neural network. *J. Chem. Soc., Perkin Trans. 2* **1991**, 1755–1762.

(15) Melssen, W. J.; Smits, J. R. M.; Rolf, G. H.; Kateman, G. Two-dimensional mapping of IR spectra using a parallel implemented self-organising feature map. *Chemom. Intell. Lab. Syst.* **1993**, *18*, 195–204.

(16) Weigel, U.-M.; Herges, R. Automatic interpretation of infrared spectra: Recognition of aromatic substitution patterns using neural networks. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 723–731.

(17) Meyer, M.; Weigelt, T. Interpretation of infrared spectra by artificial neural networks. *Anal. Chim. Acta* **1992**, *265*, 183–190.

(18) Ricard, D.; Cachet, C.; Cabrol-Bass, D. Neural network approach to structural feature recognition from infrared spectra. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 202–210.

(19) Tanabe, K.; Tamura, T.; Uesaka, H. Neural network system for the identification of infrared spectra. *Appl. Spectrosc.* **1992**, *46*, 807–810.

(20) Smits, J. R. M.; Schoenmakers, P.; Stehmann, A.; Sijstermans, F.; Kateman, G. Interpretation of infrared spectra with modular neural-network systems. *Chemom. Intell. Lab. Syst.* **1993**, *18*, 27–39.

(21) Est, Q. C. v.; Schoenmakers, P. J.; Smits, J. R. M.; Nijssen, W. P. M. Practical implementation of neural networks for the interpretation of infrared spectra. *Vib. Spectrosc.* **1993**, *4*, 263–272.

(22) Bourne, S.; Reedy, G. T.; Cunningham, P. T. Gas chromatography/matrix isolation/infrared spectroscopy: An evaluation of the performance potential. *J. Chromatogr. Sci.* **1979**, *17*, 460–463.

(23) Reedy, G. T.; Bourne, S.; Cunningham, P. T. Gas chromatography/infrared matrix isolation spectrometry. *Anal. Chem.* **1979**, *51*, 1535–1540.

(24) Reedy, G. T.; Ettinger, D. G.; Schneider, J. F.; Bourne, S. High-resolution gas chromatography/matrix isolation infrared spectrometry. *Anal. Chem.* **1985**, *57*, 1602–1609.

(25) Bourne, S.; Reedy, G.; Coffey, P.; Mattson, D. Matrix isolation GC/FTIR. *Am. Lab. (Fairfield, Conn.)* **1984**, *16*, 90–101.

(26) Baba, N. A new approach for finding the global minimum of error function of neural networks. *Neural Networks* **1989**, *2*, 367–373.

(27) Yu, X.-H.; Cheng, S.-X. Training algorithms for backpropagation neural networks with optimal descent factor. *Electron. Lett.* **1990**, *26*, 1698–1700.

(28) Alder, M.; Lim, S. G.; Hadingham, P.; Attikiouzel, Y. Improving neural net convergence. *Pattern Recognit. Lett.* **1992**, *13*, 331–338.

(29) Herrmann, L.; Rienäcker, U. Verbesserung von Lernverhalten und Diskriminationsleistung neuronaler Netze bei der Mustererkennung in Biosignalen (Improved learning capacity and discrimination performance of neural networks in pattern recognition of biosignals). *Biomed. Tech.* **1992**, *37*, 69–72.

(30) Park, D.-J.; Jun, B.-E.; Kim, J.-H. Novel fast training algorithm for multilayer feedforward neural network. *Electron. Lett.* **1992**, *28*, 543–545.

(31) Joya, G.; Frias, J. J.; Mar Martin, M.; Sandoval, F. New learning strategies from the microscopic level of an artificial neural network. *Electron. Lett.* **1993**, *29*, 1775–1777.

(32) Leonard, J.; Kramer, M. A. Improvement of the backpropagation algorithm for training neural networks. *Comput. Chem. Eng.* **1990**, *14*, 337–341.

(33) Troll, A.; Feiten, W. Improving neural net training by mathematical optimization. *Cybern. Syst.* **1992**, *23*, 447–457.

(34) Reyneri, L. M.; Filippi, E. Modified backpropagation algorithm for fast learning in neural networks. *Electron. Lett.* **1990**, *26*, 1564–1566.

(35) Prados, D. L. New learning algorithm for training multilayered neural networks that uses genetic-algorithm techniques. *Electron. Lett.* **1992**, *28*, 1560–1561.

(36) Fortuna, L.; Graziani, S.; Lo Presti, M.; Muscato, G. Improving back-propagation learning using auxiliary neural networks. *Int. J. Control* **1992**, *55*, 793–807.

(37) Barkema, G. T.; Andree, H. M. A.; Taal, A. The patch algorithm: fast design of binary feedforward neural networks. *Network (Bristol)* **1993**, *4*, 393–407.

(38) Ingman, D.; Merlis, Y. Local minimum escape using thermodynamic properties of neural networks. *Neural Networks* **1991**, *4*, 395–404.

(39) Gordon, M. B.; Pereto, P.; Rodriguez-Girones, M. Learning in feed-forward neural networks by improving the performance. *Physica A* **1992**, *185*, 402–410.

(40) Nedelković, V. A novel multilayer neural networks training algorithm that minimizes the probability of classification errors. *IEEE Trans. Neural Networks* **1993**, *4*, 650–659.

(41) Heskes, T. M.; Slijpen, E. D. T.; Kappen, P. Learning in neural networks with local minima. *Phys. Rev. A* **1992**, *46*, 5221–5231.

(42) Smith, E. G.; Baker, P. A. *The Wiswesser line-formula chemical notation (WLN)*, 3rd ed.; Chemical Information Management, Inc.: Cherry Hill, NJ, 1975.

(43) Aoyama, T.; Ichikawa, H. Reconstruction of weight matrices in neural networks—a method of correlating outputs with inputs. *Chem. Pharm. Bull.* **1991**, *39*, 1222–1228.

(44) Harrington, P. d. B. Sigmoid transfer functions in backpropagation neural networks. *Anal. Chem.* **1993**, *65*, 2167–2168.

(45) Rotter, H.; Varmuza, K. Beurteilungskriterien für Klassifikatoren zur automatischen Spektreninterpretation (pattern recognition) (Criteria for the evaluation of classifiers for the automatic interpretation of spectra (pattern recognition)). *Org. Mass Spectrom.* **1975**, *10*, 874–884.

(46) Soltzberg, L. J.; Wilkins, C. L.; Kaberline, S. L.; Lam, T. F.; Brunner, T. R. Evaluation and comparison of pattern classifiers for chemical applications. *J. Am. Chem. Soc.* **1976**, *98*, 7139–7144.

(47) Blank, T. B.; Brown, S. D. Data processing using neural networks. *Anal. Chim. Acta* **1993**, *277*, 273–287.