Chemical Reviews®

Volume 93, Number 8 December 1993

Feed-Forward Neural Networks in Chemistry: Mathematical Systems for Classification and Pattern Recognition

John A. Burns and George M. Whitesides*

Department of Chemistry, Harvard University, Cambridge, Massachusetts 02138

Received December 23, 1992 (Revised Manuscript Received June 11, 1993)

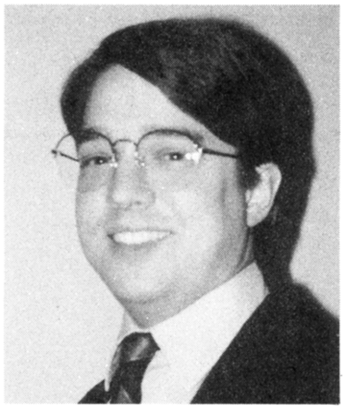
Contents

I. Introduction

Chemists have begun to use feed-forward neural networks (FFNs)[1] to solve problems. These networks can identify compounds and substructures in spectra; they can interpret sequences of proteins and DNA; they can predict properties of molecules such as reactivity or odor from molecular features; they can deduce the composition of chemical mixtures. It is our purpose to review the published work concerning the uses of FFNs in chemistry and related fields and to infer promising techniques and fields of application for this powerful new method.

FFNs have proved most useful in problems requiring recognition of patterns or classification of data (e.g., "Does this collection of structural features suggest a specified type of biological activity?"). They are especially appropriate where the relationship between input and output is complex and nonlinear, when rapid processing of information is required, when only approximate solutions are needed, and when large quantities of data broadly distributed over a wide range of examples are available. FFNs are not "programmed" in the sense used in rule-based systems; instead, they are "trained" by a process in which their internal structure adjusts empirically to obtain the best correspondence between the output of the network and the desired result for a given set of input data. In a sense, the training data and training procedures for FFNs replace the analytical methods and programming used in more conventional systems of computer-based analysis.

The applications of FFNs in chemistry are just beginning; their use in other areas of science and technology (analysis of sonar returns, voice and signature recognition, optical character recognition, vibration analysis, analysis of quality of manufactured parts) has advanced farther. Some of their most useful properties (for example, their speed in recognizing patterns) have hardly been exploited in chemistry. Nonetheless, chemistry, especially organic chemistry, relies heavily on recognition of patterns in complex, nonlinear data, using intellectual processes that are often difficult to specify analytically. FFNs have a structure that may make them particularly useful in this kind of activity.



John A. Burns received his B.A. in chemistry from Rice University in 1986. At Harvard University, he received his A.M. in chemistry in 1989 and his Ph.D. in 1993 under the direction of George M. Whitesides. His current research interests are the applications of neural networks to problems in chemistry.

George M. Whitesides received his B.A. from Harvard University in 1960 and his Ph.D. from the California Institute of Technology in 1964 with J. D. Roberts (working in NMR spectroscopy). He taught at the Massachusetts Institute of Technology from 1963 to 1982 and is now at Harvard. His research interests include materials science, biochemistry, and physical–organic chemistry.

A. Objectives of This Review

This review has four principal objectives. First, we outline the theory behind FFNs. We describe their basic structure and method of operation and attempt to clear some of the hyperbole associated with them. Second, we review the considerable body of work that has used FFNs in chemistry. Third, we evaluate the methodology currently used to apply FFNs to chemical problems. Some of the techniques being used substantially improve on the basic algorithms, while others have little sound basis. Fourth, we recommend procedures for the applications of FFNs to new problems. By examining the current state of the art, we hope to make this method accessible to more chemists.

B. The General Problem of Prediction

Many techniques, including FFNs, purport to find relationships among data and predict new values. These methods fall into two classes: the *theoretical*, in which analytical methods based on a physical model are applied to calculate an answer, and the *empirical*, in which internal consistencies in the data determine the result.[2] Theoretical models are more satisfying intellectually, but many problems either have no well-defined theoretical basis or do not allow practical computational solutions based on theory due to their complexity. In cases where theoretical methods are inadequate, empirical methods can give useful results.

A variety of ways to approximate empirical relationships have found wide application. Each of them has characteristic problems that limit its general usefulness. For instance, the behavior of a mathematical series depends strongly on its basis functions: if the functions chosen do not reflect the fundamental behavior of the system, the method may approximate the real relationship poorly, especially if only a few terms are considered. A corollary concern is the behavior of these functions at the extremes: polynomial expansions explode, and Fourier expansions oscillate with unmitigated strength outside the region of the given data. Statistical methods usually make simplifying assumptions about the relationships among data, such as independence of the input variables. These assumptions complicate the handling of nonlinear effects and complex interdependencies. The nearest-neighbor method,[3] a high-dimensional interpolation technique, depends on assumptions about the definition of "near" (i.e., the relative importance of different features and the significance of differences in each particular feature). Furthermore, as a linear method, it does not have a graceful way to handle curvature. In contrast to methods that approximate continuous functions, decision trees provide only classification, rather than continuous numerical output.[4] Decision trees draw sharp divisions. They cannot take into account conflicting effects unless they are explicitly represented.

An FFN builds up an approximation by combining the effects of several basis functions, usually sigmoids. Unlike most mathematical series, a collection of sigmoids reaches an asymptote at the extremes. The FFN approximates the function by iterative adjustment of the parameters describing the effect of one component function on another. The "training process" results in discrimination among the different inputs, automatically weighting important inputs and their combinations. The predecessor of FFNs, the perceptron method,[5] can only represent linearly separable effects. Perceptrons improve on simple statistics by taking advantage of correlations between inputs. With an internal structure added to the perceptron model, FFNs can reflect effects dependent on more than one input. One can consider FFNs "universal approximators": an FFN can in principle represent *any* function if it contains enough units.[6]

C. Background on Connectionism

FFNs form a subset of connectionist models.[7–14] Researchers invented these models in a presumed analogy to the brain, considered as a network of fundamental elements (neurons) with many connections. The models reflect the hypothesis that information in a brain resides in the strengths of connections between neurons, not in the internal state of individual neurons. Learning occurs through adjustment of the strengths of the connections and corresponds to the adjustment of the parameters in neural network models.

The field of "neural networks" has widened to include subjects from medical studies of nerve response to abstract mathematical examinations of neural network algorithms. The implementations applied in the work reviewed here are strictly mathematical models, suitable for use as tools for the approximation of functional relationships.

D. Why Use Neural Networks in Chemistry?

Many problems in chemistry lack adequate analytical tools for explanation or prediction. In every area where a chemist's intuition surpasses computational techniques, the computer method has failed somewhere to include or extract as much information as a chemist can. In some cases, the difficulty lies in defining and quantifying a property (e.g., odor). In other cases, the difficulty lies in finding good features or properties to provide to the network as input: for example, no *simple* definition or description of molecular shape exists. FFNs also appear to surpass statistical methods on problems with noisy or erroneous data.[15-17] This class of problems forms one group of potential applications of FFNs.

FFNs can also prove useful in certain problems that existing methods have solved, provided that the network solves the problem more rapidly or less expensively than conventional methods. Although the process of training a network to predict a given property usually takes a long time, a trained network makes predictions very quickly—in some cases, fast enough to make new applications possible. For example, Gezelter and Freeman used FFNs to generate NMR pulse shapes quickly enough, in principle, to design a new pulse shape during an experiment.[18]

The power of FFNs has been increased by combining them with other methods. One can feed the output of the network into a second process, such as a molecular mechanics computation in the case of a predictor of protein structure.[19] Techniques such as principal component analysis[15,20] or information analysis[21] can provide preprocessed input to the network. FFNs can also be used in parallel with other neural networks (or other methods) to generate consensus predictions.[22-24]

II. How Feed-Forward Neural Networks Work

A. General Description

A hierarchy of "units" built up into a complex function comprise an FFN.[25] Each unit generates as output a simple function of its inputs, which may include external data or the outputs of previous units. Each unit takes the output of previous units as input and provides its output as input to subsequent units (hence the term "feed-forward"). The coefficients that multiply the inputs to a unit are called "weights". These coefficients adjust to make the network "learn" its training data. Although the individual units are simple, the composite function can be quite complex.

The internal architecture of an FFN looks like a flowchart (Figure 1). Most FFNs are organized in layers: each unit in a layer takes all the units in the previous layers as inputs, performs a calculation, and provides its output as input to all the units in the next layer. This method of connection is not obligatory. The units are sequential: each unit can take any or all previous units as input and provide its output as input to any or all subsequent units.

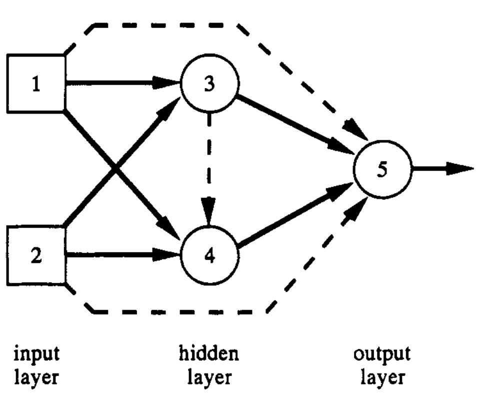


Figure 1. This flowchart represents a fully connected layered network with two inputs (1 and 2), two hidden units (3 and 4), and one output (5). Each unit is connected to each of the units in the adjacent layers, as shown by the solid lines. A network fully connected without regard for layers would include the connections between distant layers (1–5, 2–5) and within layers (3–4), shown here as dashed lines. Because input units do no calculation (they represent input values), a connection between input units 1 and 2 would be meaningless.

The units in the first layer represent the external inputs to the network and do no calculation. The outputs of the units in the last layer are the outputs of the network as a whole and represent the "answer". Because the units in layers between the input layer (first) and the output layer (last) do not correspond to external inputs or outputs of the network, they are called "hidden units".

In this paper, we express the architecture of a particular layered FFN as a series of numbers separated by colons. The notation 80:8:4 represents a fully-connected layered network with 80 units in the input layer, corresponding to 80 external inputs; 8 hidden units in a single middle layer; and 4 output units, corresponding to four output values of the network. In some cases, the input values originated in a matrix; for example, a set of inputs corresponding to each natural amino acid (20) at each of 4 positions in a polypeptide would give rise to a 4 by 20 input matrix, which can also be considered an 80-valued vector. For clarity, we give the dimensions of the matrix rather than the total in our description, e.g., 4·20:8:4 for a network with the same architecture as the previous example in which the input values come from a 4 by 20 matrix. The FFN does not distinguish between matrix input and unrelated input; the references are for human convenience only.

B. Mathematical Description of Standard Network

The standard mathematical implementation of an FFN[7] has a hierarchy of identical units. [The following description is supplemented by eqs 1–3 (Chart I) and Figure 2.] The unit multiplies each input by the weight of the connection between itself and the unit providing the input. It adds the weighted inputs and a bias term together (eq 1). The bias term can be regarded as the weight of an input fixed at 1 (eq 1a).[26] The unit applies the sigmoid function to the sum of the weighted inputs (eq 2). That function can resemble linear, step, or exponential functions on different scales and in different

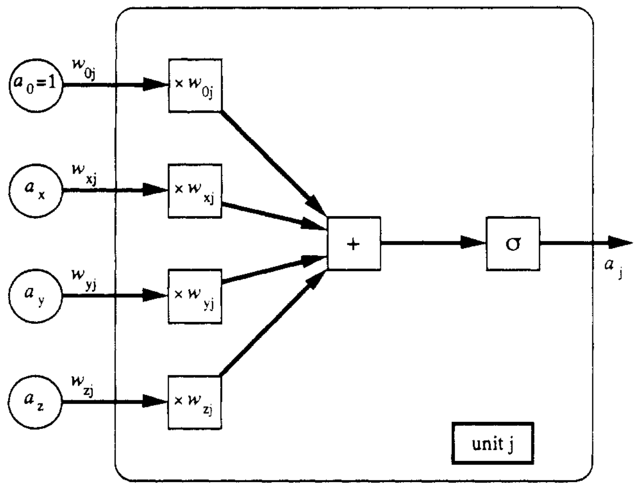


Figure 2. The mathematical function of a unit can be considered a three-stage process: (1) The unit multiplies each input by the weight for that particular connection. The weights can be positive or negative in sign. The bias is treated as a weight applied to an input of constant 1, as shown in eq 1a. (2) The unit adds the weighted inputs together. (3) The unit applies the sigmoid function to the sum with the formula in eq 2.

parts of its domain (Figure 3). Since the component units are explicit functions, their composite, the FFN, can be written as an explicit function as well (eq 3). One can regard the outputs of the hidden units as intermediate values in a calculation. Their form makes these calculations, and modifications of the network, as simple as possible.

Chart I

For a given unit j, a_j is the output, b_j is the bias, w_{ij} is the weight connecting it to unit i, and S_j is the weighted sum of its inputs. This expression does not assume layering: every previous unit is considered an input. If two units are not connected, the weight between them is fixed at zero.

$$S_j = b_j + \sum_{i=1}^{j-1} a_i w_{ij} \quad (1)$$

The bias can also be expressed as an input $a_0 = 1$ with a weight $w_{0j} = b_j$.

$$S_j = a_0 w_{0j} + \sum_{i=1}^{j-1} a_i w_{ij} = \sum_{i=0}^{j-1} a_i w_{ij} \quad (1a)$$

$$a_j = \sigma(S_j) = 1/[1 + \exp(-S_j)] \quad (2)$$

For the network in Figure 1:

$$w_{15} = w_{25} = w_{34} = 0$$

$$\text{net output} = a_5 = \sigma(b_5 + a_3 w_{35} + a_4 w_{45}) = \sigma[b_5 + w_{35}\,\sigma(b_3 + a_1 w_{13} + a_2 w_{23}) + w_{45}\sigma(b_4 + a_1 w_{14} + a_2 w_{24})] \quad (3)$$

The process of training a network consists of adjusting the weights to minimize disagreement between the output of the network and desired values for a set of training examples (the "training set") with known correct outputs. An error function, usually defined as the total squared error (eq 4), quantifies the disagreement. The derivative of the error function with respect to each weight is determined using the input and output values of one example. One advantage of the sigmoid function used is that it is straightforward to calculate the derivative of the output of a unit with respect to the weight of an input (eqs 5–7).[7]

$$E = {}^1/_2 \sum_{\text{outputs}} (y_n - a_n)^2 \quad (y \text{ is desired value}) \quad (4)$$

$$\mathrm{d}\sigma/\mathrm{d}x = \sigma(x)[1 - \sigma(x)] \quad (5)$$

$$\frac{\mathrm{d}S_j}{\mathrm{d}w_{ij}} = \frac{\mathrm{d}}{\mathrm{d}w_{ij}} \sum_{n=0}^{j-1} a_n w_{nj} = a_i \quad (6)$$

$$\frac{\mathrm{d}a_j}{\mathrm{d}w_{ij}} = \frac{\mathrm{d}a_j}{\mathrm{d}S_j}\frac{\mathrm{d}S_j}{\mathrm{d}w_{ij}} = \frac{\mathrm{d}\sigma(S_j)}{\mathrm{d}S_j}\frac{\mathrm{d}S_j}{\mathrm{d}w_{ij}} = a_i a_j(1 - a_j) \quad (7)$$

The adjustment to the weights is often simple gradient descent: each weight adjusts by a small amount proportional to the derivative of the error function with respect to that weight, and in the opposite direction (eq 8).

$$\Delta w_{ij} = -\epsilon(\mathrm{d}E/\mathrm{d}w_{ij}) \quad (8)$$

The "learning rate" ϵ is usually set by the researcher. Because of the complexity of the function described by the network, the error function often fluctuates over small changes in the weights. As the learning rate gets larger, the weights change more quickly, but the danger that the weights will overshoot desirable values increases. Learning rates vary from problem to problem, usually in the range 0.001 to 1.

The introduction of an averaging term improves on strict gradient descent.[7] Each new calculated step down the gradient (eq 8) includes a fraction of the previous step, which itself includes a fraction of the step before, and so on (eq 9).

$$\Delta w_{ij}\ (\text{iteration } n) = -\epsilon(\partial E/\partial w_{ij}) + \mu \Delta w_{ij}\ (\text{iteration } n - 1) \quad (9)$$

The coefficient μ of the previous step is referred to as the "momentum" because it opposes a change in direction between successive steps of minimization, as inertia opposes a change in direction of physical motion. As the momentum increases, so does the ability of the minimization process to deal with fluctuations in the gradient of the error due to changing input patterns or conflicting examples. The momentum term can also carry the weights past a local minimum. The network changes the direction more slowly in response to significant changes in the gradient of the error. Values of the momentum term are usually set at 0.5 to 0.9; smaller values have little stabilizing effect, while values greater than 1 do not allow the effects of previous steps to decay.

One can organize the training process in one of two ways. If one adjusts the weights on the basis of the error in each particular example, the process will get "stuck" less often at a local minimum where different examples cancel out because the direction of the adjustment of the weights changes from example to example. Alternatively, one can adjust the weights on the basis of the error of all the examples. This process assures that the overall error always forms the basis for adjustment of the weights. It also stops different examples from causing adjustments in opposite directions. That prevention can be an advantage if it reduces the amount of wandering, and therefore speeds the training of the network; it can be a disadvantage if it stalls the training process. As a compromise between the two methods, one can adjust the weights each time on the basis of some, but not all, of the examples.

C. Modifications of Calculations

Often, the network can be trained efficiently by beginning with a large learning rate and reducing it over the course of training. It can skip around the possible combinations of weights and find a region of relatively low error before the step sizes become small enough to be affected by small fluctuations in the gradient. One can decrease the learning rate in a predetermined fashion,[27] relate the learning rate to the change in the error on each step,[28] or reduce the learning rate if the change in weights overshoots enough to increase the overall error.[29] Each unit can have a different learning rate and momentum.[30]

Some researchers have used slight variations on the standard back-propagation method. One can eliminate the sigmoid function from the output units, and simply take the sum S_i instead (eq 1). The advantage of using this method is that the output of the sigmoid function is constrained to the range 0 to 1; that restriction does not apply to a sum. The disadvantage is that the absence of the sigmoid function and its ability to fit several different functions removes a degree of adaptability from the network, so the network might need more hidden units to reach the same accuracy or quality of fit.

One very common variation speeds the learning process by applying a trainable coefficient α (the "gain"), sometimes expressed as $1/T$) to the sum in evaluating the sigmoid (eq 10).[31]

$$\sigma'(S_j) = \frac{1}{1 + \exp(-\alpha S_j)} = \sigma(\alpha S_j) = \sigma(\sum_{i=0}^{j-1} a_i(\alpha w_{ij})) \quad (10)$$

Because this term multiplies the weights, which can assume any value without the coefficient anyway, it does not change the possible states of the network. It does affect the magnitudes of derivatives of the error, and thereby the path of the training process.[32]

Some researchers adjust α by arbitrary methods because they misunderstand its effect. When they do, the weights must compensate for the change in α, which may slow learning. Other counterproductive modifications of the basic unit include the combination of linear terms with sigmoid terms in the same unit and the "unsaturated" neuron.[33] These formulations reveal a failure to understand the flexibility of the sigmoid function (Figure 3).

The error function can represent the researcher's definition of optimal fit better than the standard function (eq 4) does. One can ignore small errors on particular examples when they fall below an acceptable amount.[34] To prevent outliers from unduly influencing training, one can truncate the derivative of the error function at a maximum value.[34] To emphasize particular examples, one can magnify the errors made in

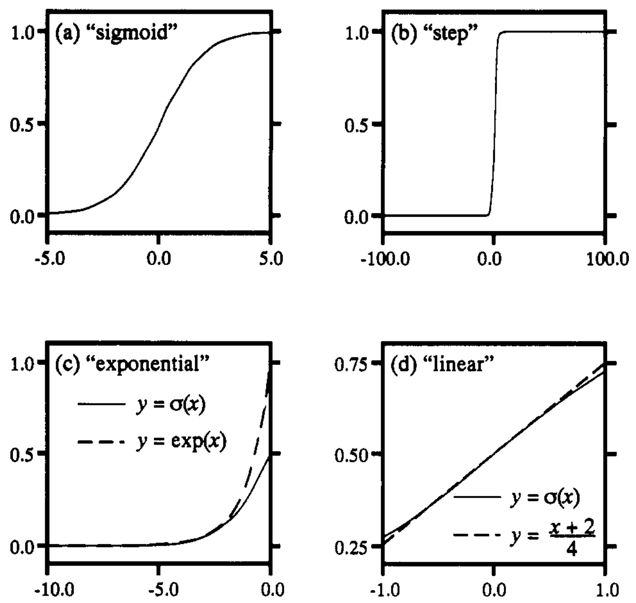


Figure 3. The sigmoid function (eq 2) can be scaled and positioned to resemble other functions over part of its domain: (a) in the range −5 to 5, the function has a typical sigmoid appearance; (b) over the range −100 to 100, the sigmoid resembles a step function; (c) the sigmoid approximates the function e^x closely up to about −2; (d) The central part of the function (−1 to 1) is nearly linear.

those outputs to cause a correspondingly larger adjustment. In all of these cases, the researcher tailors the error function to reflect the particular goal of the application rather than an arbitrary standard practice.

Another consideration often added to the minimization process is the removal of small effects. To decrease the effect of small weights, one can systematically reduce them[34] or periodically reset small weights to zero.[35] In either case, if they do not matter, they will tend to remain close to zero after being reduced. One can also remove units: units with small input weights (indicating that they extract no information) or small output weights (indicating that they are ignored) can be eliminated from the network gradually, by weight reduction, or suddenly, by removal of their connections.[34] These modifications prevent overtraining through small quirky adjustments by excess units. They also bias the process to produce networks with only big effects, making them easier to interpret as "rules". Such a result could be good or bad, depending on whether such large effects really exist. Like Occam's razor, this procedure produces the simplest answer, not necessarily the correct one.

Other methods to train neural networks exist. Three other gradient-based methods can be considered close relatives of back-propagation. The conjugate gradient is common in other work,[36] but few chemists have so far applied it.[37] One also can treat the minimization as a problem in differential equations.[38–40] The variable metric has also provided good results.[41–44] Two methods do not use the gradient. Genetic algorithms alter weights in a population of neural networks by exchange and mutation and then screen out poor performers, in an analogy to biological evolution.[45] In simulated annealing, the weights change randomly, with the random movements more favored if they do not increase error.[46] This method can also assist back-propagation: by applying random variations when the directional method gets stuck, one can keep the minimization process moving until it arrives at a good local solution (i.e., one better than any other with similar weights).

D. Evaluating the FFN

The process of training a neural network optimizes results on training examples. With appropriate values for the learning rate and momentum, its performance on training examples will continually improve with successive training iterations. In contrast, performance on new examples tends to reach a maximum, then degrade as the network "overtrains" to reach higher accuracy on the training examples. Similarly, a network with too many hidden units can easily fit the training data without forming a good generalization. Since the real criterion for the success of the network is its ability to predict new examples, that criterion should be used to select the architecture and training time of the network also. When the number of examples is too small to allow reserving a representative test set, one can use cross-validation to generate useful estimates of predictive ability.

Cross-validation consists of dividing a set of examples into subsets and using each subset in turn as a test set for a network trained on the remainder of the examples, much like bootstrapping statistics.[47] The averaged or consensus performance on test sets gauges the quality of prediction by the network; the variability of the results among the different trials gives an indication of the sensitivity of that assessment to the exact composition of the test and training sets. If one uses cross-validation to choose the architecture and training time of the network, the consensus performance may overstate the ability of the network, if the best performance in the trial is a statistical fluke. One can reserve a test set separate from the cross-validation trial to test prediction. Often, however, several architectures and training times generate near-optimal results, reducing the chance of statistical artifacts.

Some researchers have used empirical rules to select architectures and training times. Because these rules depend on implicit assumptions about the information content of the training data, they do not necessarily perform well. Arbitrary criteria for terminating the training process include a small overall error, a small gradient of the error function, or a set of number of iterations. Several rules of thumb for designing the architecture of the network exist. Usually, only one layer of hidden units is used, although a few people use two.[48] The network is more efficient if the output units share hidden units, as in standard layering, rather than maintaining hidden units unconnected to other output units.[34] One rule sets the number of weights to a small multiple of the number of training examples.[49-51] Many researchers choose a number of hidden units somewhat intermediate between the sizes of the input and output layers;[52,53] most do not discuss or defend their reasons for choosing a particular architecture.

E. Modification of Input Data

The particular form in which properties are presented to the network matters. Units explicitly sum, making addition a "natural" function for the network to model. If the network needs to multiply two values, then one should use the logarithms of those properties as input rather than the properties themselves. Adding the logarithms represents the multiplication step. Otherwise, the network would require more units to perform adequately because it would have to simulate the multiplication function as well as the rest of the relationship.

Another practical consideration is the range of the input values. In one common technique, normalization, each external input has the same range, usually 0 to 1. Otherwise, an input that has instances over a wide range will have a larger effect than an input distributed over a smaller range in the initial stages of training. The disproportionate effect distorts the training process, even though the network can compensate by reducing the weights for an input with a large range. (If some inputs are known or assumed to be more important than others, such distortion could be desirable.) Because the output of a sigmoid also has the range 0–1, one can compare the weights assigned to normalized external inputs in the same way as those assigned to hidden units, although such interpretation is nonetheless difficult.

Computer-generated data appear precise. Because the network has no way to guess a reasonable degree of error, it may generate an unnecessarily complicated function to achieve high precision (overfitting). Anyone who has fitted a polynomial function to a small, noisy data set will recognize the problem. One can reduce this problem by adding small random values (noise) to the input or output values.[34] If the inputs or outputs represent sequential values, one can shift the alignment between the values from the example and the inputs of the networks.[54] In either case, zero error becomes impossible due to the movement of the data points, so the effect of the training process diminishes substantially after the error falls within the limits of the noise. This process also helps prevent local minima from impeding the training process because the error function changes slightly with every addition of new random noise. To achieve a relatively precise fit, one can reduce the amount of noise slowly over the course of training to allow close fitting after the network has had the opportunity to approach a good general fit.

When input values all represent the same kind of property, input values can represent background noise. The network can use the "noise" inputs to gauge whether the values in the "signal" inputs are significant. The network learns that it should disregard values of the signal inputs comparable in size to the noise inputs. A neural network that identifies peaks in IR spectra used this technique to good effect.[24]

Although an FFN can theoretically model any function. preprocessing the input in ways known to make it more informative reduces the computational load on the network.[49] Some researchers have provided the products of the raw inputs to the network. This technique is widely used in statistical analysis to capture effects that depend on more than one input. When the inputs are binary (0 or 1), multiplication of the inputs lends itself to a special interpretation. Because the product of binary values is 1 only if the individual values are all 1, the product used as input reflects the presence of the *exact combination* of its constituent raw inputs.

F. Using FFNs for Classification

Although in theory an FFN of sufficient size can model any relationship to any degree of precision, the fact that the network approximates a relationship from a limited number of examples with a finite number of units makes it more prudent to interpret the output as qualitative rather than quantitative. The simplest way to make that approach rigorous is to classify each example on the basis of the range in which its outputs fall, rather than to make use of the numerical value of the output itself.

When the output of the network is a classification, one must convert the continuous value of a unit to a binary value representing a yes-or-no answer. The simplest way rounds off, that is, uses 0.5 as the dividing value. One can regard intermediate values of a classifying output as probabilities or confidence levels of a "yes".[34,35] One can also choose the dividing line that gives the best results on the training set. The most stringent criterion sets separate thresholds close to 0 and 1 for binary values, with values between the thresholds considered "unknown" or "undecided". Although this method sacrifices some correct answers, it identifies examples for which the network does not generate clear results.

In many problems, the output classification consists of several mutually exclusive categories (for example, the structural types α-helix, β-sheet, and random coil at a particular position in a protein). The usual way to represent this situation is to assign one output unit for each possible category. The network "chooses" the category corresponding to the output with the highest value. Despite its elegance, this method has one principal weakness: the competitive nature of this paradigm places less common categories at a disadvantage.

A network starting from a random set of weights can reduce error in predicting a rare category by always predicting that the example is not in that category. The network can also increase apparent precision by improving the prediction of a common class slightly rather than improving the prediction of a rare class greatly. To compensate for these effects, one can present examples from rarer classes more often. One method to even out different frequencies is to present each class with the same frequency, or to divide the error attributable to each example by the proportion of its class in the training set. Another method is to adjust the ratio of examples from different classes to optimize the predictive ability of the network. Alternatively, one can automatically increase the influence of poorly predicted classes, rather than rare classes, in the training set.[56] For classes with few examples, sampling error still causes a problem even if underrepresentation no longer does.[57]

In these problems, the total squared error does not appropriately measure the quality of prediction: it takes no account of correct categorization. One can count the fraction of examples that the network predicts correctly. One researcher used the product of the fraction of correct answers for each class.[28] These values behave badly from a statistical standpoint. A result of 0 corresponds to generating the wrong answer every time, a statistically important result.

A more statistically robust measure of the quality of classification is the correlation coefficient r (eq 11).[58]

$$r = \frac{pn - uo}{\sqrt{(n+u)(n+o)(p+u)(p+o)}} \tag{11}$$

p = number of correct positive predictions

n = number of correct negative predictions

o = number of incorrect positive predictions

u = number of incorrect negative predictions

A correlation coefficient of 0 means the results cannot be distinguished from chance. A network classifying perversely would have a negative correlation. In contrast to the other measures, the correlation coefficient applies only to two-category divisions. One can assess problems with more categories by taking the correlation coefficient of particular dividing lines individually. For example, for a problem classifying α-helix, β-sheet, and random-coil structures in a protein, one would measure the correlation for predicting helix vs sheet and coil together, coil vs helix and sheet together, and sheet vs helix and coil together. Since the qualities of those classifications probably differ, the individual correlations provide more information than an overall success rate. This criterion can provide an error function for the training process called "mutual information".[59]

III. Applications of Neural Networks in Chemistry

A. Representation of Chemical Information

A researcher studies physical or abstract objects that have particular meanings, relationships, and interpretations. From a mathematical point of view, each input to and output from a layered FFN is an independent and equivalent real number. In some problems, one can reduce the information about each example or object to a list of numbers in an obvious way. In others, finding a good representation may require more effort than applying the network. We have organized this review into fields that use a common representation of input data, since the function of the network may be understood differently for the different cases. Each subsection begins with a description of the representation used in that field.

In the first category of applications, interpretation of biochemical sequences, the gene or protein contains a sequence of discrete elements. Similarly, a spectrum is made up of a sequence of intensities, which are real numbers rather than individual items. The outputs of a sensor array are real values representing similar properties, but without any particular order. In each of these three cases, the phenomena being studied have an obvious numerical representation. That set of data may be preprocessed to extract information known to be useful, but the basic representation remains the same. Various descriptive methods have been applied to represent chemical structures as lists of numbers for FFNs, since no obvious or clearly superior method exists.

$$
\begin{array}{cc|cccccccccc|l}
 & & \multicolumn{10}{c}{\text{position}} & \\
 & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \\
\hline
\text{base} & G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \\
 & C & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \text{input} \\
 & A & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \text{matrix} \\
 & T & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \\
\hline
\text{sequence} & & T & A & C & T & C & A & T & G & A & G & \\
\end{array}
$$

Figure 4. A 4 by 10 matrix of binary values represents a 10-base sequence of DNA. Each column represents a position in the sequence. Each row represents one of the four bases. The input corresponding to the base present at a given position is 1; the rest are 0. Overall, the FFN has 40 inputs in this example.

B. Applications in Biological Sequences (Sequential Binary Data)

Biochemists and molecular biologists wish to interpret biochemical sequences (proteins and DNA) in order to understand their physical properties and their information content. As automatic sequencing methods have improved, the influx of sequence information has outpaced its interpretation. Statistical methods have been applied with some success to that task.[60,61] Generally, such methods work best when the inputs are independent and grow more complex and less reliable when they must take effects of interdependence into account. Molecular mechanics approaches have had some success in predicting the folding (shape) of proteins on the basis of their chemical structure.[62] This computer-intensive method requires a detailed representation of the molecule and a complex mathematical model of its dynamics. A corresponding model for analyzing genetic sequences would require a detailed representation of the relevant biochemical machinery of the cell as well as that of the gene itself.

To represent a set of discrete elements, such as the four bases in DNA or the naturally occurring amino acids, one can provide as input a checklist of all possible items in the set. Inputs representing combinations of items have also been used.[35,50] An input corresponding to the base present in the example has the value 1; the rest have the value 0. This system is called "sparse coding" because most of the inputs are 0. For an input corresponding to 10 base pairs in DNA, the input would consist of a matrix of 10 columns, corresponding to the positions, by four rows, corresponding to each of the four bases. (See the example in Figure 4.) The input values are organized as a matrix for human convenience; the network treats each input independently. Because the inputs to the network are binary in these applications, any "relative importance" or "property" information resides in the weights. In one simple case, a network generated weights during training that correlated well with empirically generated properties.[63] Usually, especially with many hidden units, the "properties" or "features" embodied in the weights do not correspond exactly to recognizable physical quantities, but the information can nonetheless prove useful.[64]

1. Protein Structure from Sequence

A number of researchers have independently applied FFNs to the prediction of structure and function of proteins (Table I).[65,66] Most of them used a sparse coding scheme to represent a segment of the amino acid sequence.

Several groups have predicted secondary structural types (α-helix, β-sheet, or random coil) of segments of proteins. Their overall results were similar, but each group considered particular issues. The Bohr group found that homology between the sequences of training and test examples improved the predictive power of the network.[67] Holley and Karplus examined the effect of hidden units and found that two sufficed.[55] They also discovered that predictions with large output values were more likely to be right than those with smaller outputs. Qian and Sejnowski found that eliminating the layer of hidden units did not degrade performance significantly.[35] They improved the performance of their method somewhat by training a second FFN to predict the secondary structure from the outputs of the first network for the same sequence. The second network eliminated small gaps in contiguous regions of one structural type.

Kneller, Cohen, and Langridge increased the information provided to the network by computing a "hydrophobic moment" (which represents the tendency of hydrophobic groups to align in either a sheet or helix structure) of the sequence used as input.[68] This input improved the correlation coefficients of the test set slightly. They also separated the proteins into three groups: those with structures that were mostly α-helix, those mostly β-sheet, and those of mixed or random-coil structure. Networks trained and tested on these relatively homogeneous groups performed better than networks applied to broader sets of proteins, largely by ruling out less common structural types.

McGregor, Flores, and Sternberg used an FFN taking sparsely coded amino acids as input to predict the category of β-turn (type I, type II, unspecified, or nonturn) of a sequence of four residues.[63] It is unclear why they did not use a longer sequence to include the "context" of the putative turn. Because their data originally contained far more nonturns than turns, they improved prediction by increasing the ratio of turns to nonturns in their training set. In the network without hidden units, the weights of the amino acids correlated well with the empirical parameters for residue propensities derived by Chou and Fasman.[69]

The ability of FFNs to improve on first-order statistics in the prediction of protein secondary structure from the local sequence may be limited.[70] Kneller, Cohen, and Langridge achieved 65% prediction in their work;[68] on the same data set, Stolorz, Lapedes, and Xia found that one could achieve 61% by combining independent probabilities for the amino acids in the local sequence (first-order Bayesian statistics), ignoring all higher-order effects that would depend on more than one residue.[59] The method of Viswanadhan, which combined FFNs with other algorithms, improved only the prediction of the β-sheet structural type.[22] The best results come from an FFN used to combine the usual FFN method with statistical and reasoning methods.[23]

The Bohr group extended their method by including information about tertiary structure among the outputs of the network.[19] The outputs consisted of the distances between the central residue and each of the 30 residues before it, and the propensity of the residue to adopt

Table I. Applications of FFNs in Analyzing Proteins from Their Sequences

Secondary Structure (α-Helix, β-Sheet, Random Coil)

input	output	type[a]	architecture[b]	%[c]	r_α[d]	r_β	r_r	ref
sparsely coded amino acid sequence	secondary structure (α/not α)	B	51·20:40:2	73	0.38			67
sparsely coded amino acid sequence, all one secondary structure type	secondary structure (α/not α)	B	200:1	78	0.52			70
sparsely coded amino acid sequence	secondary structure (α/not, β/not)	B	17·21:2:2	63	0.41	0.31	0.41	55
sparsely coded amino acid sequence	secondary structure (α, β, random)	C	13·21:3	63	0.35	0.29	0.38	35
output of above network	secondary structure (α, β, random)	C	13·3:3	64	0.41	0.31	0.41	35
as above + hydrophobic moments	secondary structure (α, β, random)	C	275:3/13:3	65	0.42	0.32	0.43	68
as above, on mostly α-helix proteins	secondary structure (α, β, random)	C	275:3/13:3	82	0.65	0.00	0.63	68
as above, on mostly β-sheet proteins	secondary structure (α, β, random)	C	275:3/13:3	73	0.00	0.51	0.51	68
as above, on mixed α-helix/β-sheet	secondary structure (α, β, random)	C	275:3/13:3	70	0.47	0.44	0.48	68
sparsely coded amino acid sequence	secondary structure (α, β, random)	C	13·20:3	59	0.36	0.28	0.38	59
output of above network	secondary structure (α, β, random)	C	13·3:3	61	0.40	0.27	0.39	59
amino acid sequence	secondary structure (α, β, random)	C	Bayesian statistics	61	0.35	0.27	0.36	59
secondary structure outputs of FFN, statistical method, and memory-based reasoning method	secondary structure (α, β, random)	C	13·3·3:30:3	66	0.47	0.39	0.43	23
includes method from ref 27 among 6 combined methods					0.41	0.47	0.41	22
residue α-helix tendencies	force constant for helix formation	X	17·5:1	model results in good secondary and tertiary structure				e
amino acid composition, mw, presence of heme	overall fraction of secondary structure types	X	22:8:2	more accurate than counting individual FFN predictions or MLR[f]				37
fraction of secondary structure type	fraction of other secondary structure types	X	various	some useful relationships				g

Secondary Structure (β-turns: I, II, nonspecific, nonturns)

input	output	type	architecture	%	$\%_I$[h]	$\%_{II}$	$\%_{ns}$	ref
sparsely coded amino acid sequence	β-turn type (I, II, ns, not)	C	4·20:8:4	26	69	81	36	63
sparsely coded amino acid sequence	β-turn type (I, II, ns, not)	C	4·20:8:4	29; 56% within 2 residues				56

Tertiary Structure

input	output	type	architecture	results	ref
sparsely coded amino acid sequence	dist. $C_{\alpha,x}$ to $C_{\alpha,x+(1-30)}$, secondary	B	61·20:300:33	after minimization, result resembles X-ray structure	19
hydrophobicities of amino acids	distance matrix	X	140:90:140·140	correct tertiary structure for homologous proteins	71
hydrophobicities of amino acids	distance matrix	X	140:15:140·140	no generalization	72

Other Properties

input	output	type	architecture	results	ref
sparsely coded amino acid sequence	homology to protein family	B	12·20:10:1	better than statistical method PROFILESEARCH	i
sparsely coded amino acid sequence	four subregions of Ig	C	5·20:8:4	when subregions combined, fp = 7.3%, fn = 1.8% (mice), 6.2% (human)[j]	k
sparsely coded amino acid sequence (N-terminal)	cytosolic peptide or signal sequence	C	20·20:3:2:2	improved on statistical methods	l
properties and secondary structure of amino acids in sequence	water binding	B	not reported	$r = 0.14$ on test set	m
properties and secondary structure of amino acids in sequence	water binding at each atom	B	not reported	$r = 0.09$ on test set	m

[a] "B" means the output is converted to a binary classification by using threshold values; "C" means the example is classified as the category corresponding to the largest output unit; "X" means the output is interpreted as a real number or continuous property. [b] The architecture of the network is abbreviated as described in the text. [c] The percentage of test examples correctly predicted. [d] r_α, r_β, and r_r refer to the correlation coefficient of predictions of α-helix, β-sheet, and random-coil structure, respectively. [e] Head-Gordon, T.; Stillinger, F. H. *Biopolymers* **1993**, *33*, 293–303. [f] MLR = multiple linear regression. [g] Pancoska, P.; Blazek, M.; Keiderling, T. A. *Biochemistry* **1992**, *31*, 10250–10257. [h] $\%_I$, $\%_{II}$, and $\%_{ns}$ refer to the percentage of correctly classified type I, type II, and nonspecific turns, respectively. [i] Frishman, D.; Argos, P. *J. Mol. Biol.* **1992**, *228*, 951–962. [j] fp = false positive; fn = false negative. [k] Bengio, Y.; Pouliot, Y. *Comput. Appl. Biosci.* **1990**, *6*, 319–324. [l] Ladunga, I.; Czakó, F.; Csabai, I.; Geszti, T. *Comput. Appl. Biosci.* **1991**, *7*, 485–487. [m] Wade, R. C.; Bohr, H.; Wolynes, P. G. *J. Am. Chem. Soc.* **1992**, *114*, 8284–8285.

α-helix, β-sheet, and random-coil conformations. After training the network, they used one protein as a test example. Although the structure generated (after refinement by molecular mechanics calculations) resembled that found by X-ray crystallography, the result is of uncertain significance because one of the training examples was homologous in shape.

Wilcox, Poliac, and Leibman attempted to predict tertiary structure by generating a complete distance matrix as output. In a network trained on homologous proteins, the FFN predicted the tertiary structures correctly;[71] trained on nonhomologous proteins, it

Table II. Applications of FFNs to Problems in Predicting the Activity of DNA Sequences[a]

input	output	type	architecture	results[b]	ref
sparsely coded bases and combinations	*E. coli* promoter	B	81:1	≥simple statistical method	c
sparsely coded bases	*E. coli* promoter	B	44·4:4:1	98.4% right vs 77% rule-based system	28
sparsely coded bases	*E. coli* 17-base–space promoter	B	58·4:15:1	20% fn, 0.1% fp; 30% fp for rule-based	73
sparsely coded bases	*E. coli* 16-base–space promoter	B	various		74
sparsely coded bases	*E. coli* 17-base–space promoter	B	various	3% fn, 0.2% fp	74
sparsely coded bases	*E. coli* 18-base–space promoter	B	various	21% fn, 0.15% fp	74
sparsely coded bases	human pre-mRNA splice donor	B	?	some errors mistakes in database	82
sparsely coded bases	human pre-mRNA splice donor	B	19·4:20:2	errors often mistakes in database	81
sparsely coded bases	human intron vs transcribed exon	B	301·4:200:1	$r = 0.73$	75
sparsely coded bases	human pre-mRNA splice donor	B	21·4:40:1	$r = 0.50$, 0.71 with intron info	75
sparsely coded bases	human pre-mRNA splice acceptor	B	51·4:40:1	$r = 0.36$, 0.70 with intron info	75
sparsely coded sequences	human intron vs. transcribed exon	B	?	98.4% vs 90% Bayesian statistics (ref 49)	50
sparsely coded sequences	*E. coli* intron vs transcribed exon	B	?	99.5%	50
sparsely coded sequences	human pre-mRNA splice donor	B	?	94.5%, 99% with intron info	50
sparsely coded sequences	human pre-mRNA splice acceptor	B	?	91.2%, 96% with intron info	50
dicodon frequency	human intron vs transcribed exon	B	4096:1	>99% from 60 base segment	76
dicodon frequency, 6 reading frames	human intron vs transcribed exon	B	6·4096:1	99.4% from 60 base segment	76
codon frequency	human intron vs transcribed exon	B	64:1	>99% from 90 base segment	76
codon frequency, 6 reading frames	human intron vs transcribed exon	B	6·64:1	>99% from 90 base segment	76
results of other analyses	human coding region	B	7:14:5:1	$r = 0.68$; 10% fp, < 10% fn	21

[a] Explanations for the headings are given in the footnotes to Table I. [b] fp = false positives; fn = false negatives. [c] Horton, P. B.; Kanehisa, M. *Nucleic Acids Res.* **1992**, *20*, 4331–4338.

failed.[72] Because the hidden units outnumbered the training examples, the network could have easily memorized the training set. They used the hydrophobicity of each amino acid in the sequence as input, probably insufficient to allow prediction even under ideal circumstances; a network with sparsely coded input can generate the relevant properties by inference.[35]

2. DNA Signals from Sequence

The human genome initiative and other complete-sequencing projects are generating vast amounts of DNA sequence information. As a result, learning to interpret the genetic function of these sequences has become a critical issue. Researchers have applied FFNs to separating coding and noncoding regions, identifying splice sites, and locating promoters (Table II). A sparsely coded matrix represented the sequence of nucleotides. Demeler and Zhou tried a denser coding scheme with combinations of two inputs (0,0 = A; 0,1 = T; 1,0 = G; 1,1 = C) representing bases, rather than one input for each base, but found that the sparse coding scheme gave better results.[28]

In training networks to recognize splice sites or promoters, most researchers compensated for the preponderance of negative examples by presenting positive examples more frequently than negative ones.[28,50,73,74] Brunak, Engelbrecht, and Knudsen found that the strength of splicing signals and the sharpness of the division between introns and exons were complementary: weakly predicted splicing signals tended to occur on a sharp boundary between bases predicted to be coding and bases predicted to be noncoding, and vice versa.[75]

One group attempted to capture higher-order relationships explicitly.[50] The network took products of the values in the sparsely coded input matrix as input, rather than the raw values. Third-order products from adjacent positions correspond to sparsely coded codons rather than bases. Subsequent work using the frequencies of dicodons (sixth-order) in all six possible reading frames as input achieved even better results.[76] Some researchers preprocessed the sequence completely: the network took as input not sparsely coded bases but the output of algorithms designed to find coding regions. The FFN integrated the results of the other analyses to provide an improved composite prediction.[21,77–79]

FFNs have been applied to real-world problems in identifying genetic signal sequences. Rowley and Wolf used the promoter-recognition network of O'Neill to find a promoter in a gene from *Escherichia coli*.[80] Brunak, Engelbrecht, and Knudsen were training a 19·4:20:2 network on pre-mRNA splicing sites in human genes from the database at EMBL, Heidelberg, and GenBank. They found that several sequences were consistently misclassified by the network even during training. In most cases, the library data had been transcribed erroneously or appeared to have been the result of misinterpretation.[81,82] The authors proposed that FFNs be used to screen for such errors in the future.

C. Applications in the Interpretation of Spectra (Sequential Continuous-Value Data)

To a computer, a spectrum is a sequential array of continuous-value data that approximates a function. One typically recognizes as "features" in a spectrum particular shapes or simpler functions, often as simple as a sharp peak at a given position. These recognized patterns can have absolute positions (for example, an IR absorbance at 2225 cm^{-1}, indicating a nitrile) or relative positions (for example, a peak in a mass spectrogram 28 amu lower than another, possibly indicating a loss of CO or ethylene). The features that characterize a particular molecule or substructure of interest often spread across the entire spectrum. The problem of interpreting spectra lends itself to FFN methodology because of common nonlinear effects: some regions distinguish compounds more effectively

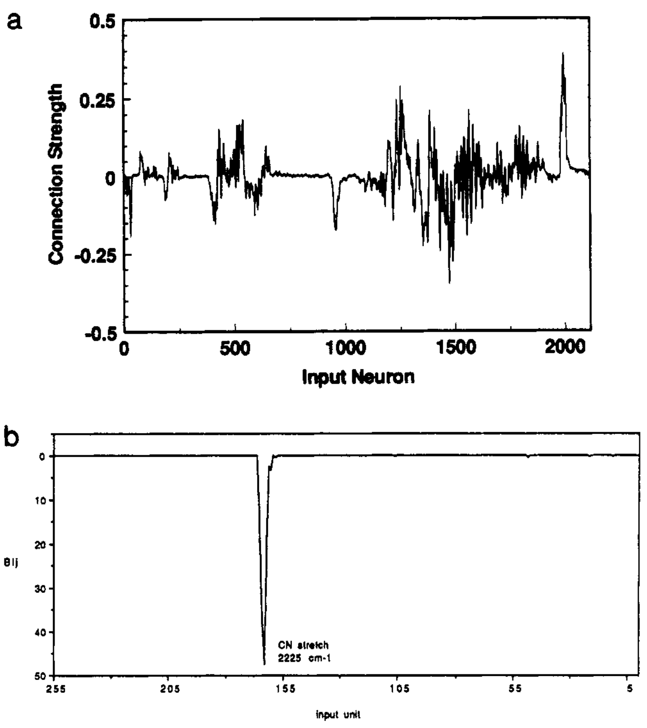


Figure 5. (a) A plot of the weights connecting the sequential inputs of a ^{1}H NMR spectrum to a hidden unit (ref 84) can be regarded as the spectrum of the feature to which the unit responds. Peaks above the lines correspond to regions where peaks in the input increase similarity; peaks below the line, to regions where peaks in the input decrease similarity. (b) A similar "feature spectrum" from ref 85 shows the weights connecting input from an IR spectrum to an output representing a nitrile group. This feature corresponds exactly to the peak a chemist would look for: the C≡N stretch. (Part A: Reprinted from ref 84. Copyright 1991 AAAS. Part B: Reprinted from ref 85. Copyright 1990 Springer-Verlag Wien.)

than others, and the significance of a feature may depend on the context.

The sequential nature of the input data allows an interesting interpretation of the patterns of weights. The ordered input values a_i approximate a function $a(x)$ of the variable used to order the data.[83] For example, in an IR spectrum, the wave number of each data point would determine its position in the sequence of inputs, and the absorbance would determine the magnitude of the corresponding input. Because the weights connecting those inputs to a unit in the next layer have the same sequence as the inputs, the weights approximate a function along the same axis (e.g., wave number). This interpretation transforms the sum in eq 1a to an integral that quantifies the overlap of the two functions (eq 12). In other words, the function w_j

$$S_j = \sum_{i=0}^{n<j} a_i w_{ij} \approx \int_0^n a(x) w_j(x) \qquad (12)$$

corresponds to the spectrum of the "feature" embodied in unit j, and the output of the unit is a measure of the similarity of the two spectra. Such "feature spectra" from a trained network have appeared in the literature (Figure 5).[54,84-87]

Networks have been applied to spectra in four different ways: to recognize them, to quantify with them, to classify them, and to transform them into a related function. In the first case, the output of the network typically represents the presence of a known substance. In the second, the output measures concentration. In the third, more refined approach, the network recognizes patterns associated with particular substructures or molecular features and identifies these structures or features as output. In the last method, the network generates a function dependent on the entire spectrum. Researchers have applied FFNs to a variety of such problems (Table III).

1. Recognition and Quantitation

A spectrum of a mixture roughly consists of the summed spectra of the pure compounds with superimposed noise. In this approximation, one measures the degree to which the spectrum of each component contributes to the spectrum of the unknown sample. An FFN has an advantage when others methods handle noise poorly, or when interactions between the components cause deviations from linear summation.

Some published work used FFNs to identify the ^{1}H NMR spectra[54,84] or MS[88] of complex pure compounds. The networks recognized new spectra and spectra to which some noise had been added, but the authors did not show superiority of the FFN method over other methods for identifying spectra. Even in the most difficult application (distinguishing oligosaccharides with the hump region of the ^{1}H NMR spectrum[84]), the most similar examples in the training set differ noticeably, suggesting that any numerical method should distinguish between them.

Quantitation of mixtures by FFNs based on IR, UV, and X-ray fluorescence spectra has outperformed other methods in published comparisons. Two groups improved the performance of their method by combining it with principal component analysis (PCA), a method in which the linear combinations of inputs having the greatest variance are extracted.[15,20] They calculated the principal components of their data set and then used the principal components as inputs instead of the raw spectrum. Two groups found that on artificially generated data sets, even with added noise, the network performed about as well as other methods; only on real data did it surpass them, suggesting that the network can take into account deviations from ideal behavior better than other methods.[15,17]

2. Transformation

An FFN can perform transformations on spectra to provide related functions, rather than single values, as output. Simple applications in this area include analyzing peaks[24,89] and extrapolating the heat capacity of polymers.[90] Sumpter has also applied FFNs to molecular dynamics.[91-93] Another potentially useful application is the design of pulse shapes for NMR experiments.[18] The Bloch equations calculate the frequency-domain excitation pattern of a nucleus from the pulse shape. No analytical method exists for the reverse calculation of pulse shape from excitation patterns. Instead, one can adjust a likely pulse shape semirandomly ("simulated annealing") in an attempt to find a pulse shape that gives a satisfactory excitation pattern.[94,95] An FFN trained to calculate the pulse

Table III. Applications of FFNs to Problems in Interpreting Spectra[a]

input	output	type	architecture	results	ref
	Recognition of Known Spectra				
^{1}H NMR spectrum	alditol identities	C	400:6:6	recognized slightly altered spectra	54
^{1}H NMR spectrum	oligosaccharice identities	C	2113:10:13	recognized new spectra	84
(structural region only)	oligosaccharide identities	C	1003:5:5	recognized new spectra	84
(hump region only)	oligosaccharide identities	C	981:5:5	recognized new spectra	84
mass spectrum	identities of sugar derivatives	C	400:25:22	claim prediction, no published results	88
	Quantitation of Known Compounds from Spectra				
IR or UV spectrum	concentration of one compound in mixture	X	various	usually beat PCA[b]	c
IR or UV spectrum	concentration of one compound in mixture	X	various	usually beat PCA	15
principal components of IR	concentration of one compound in mixture	X	various	usually beat FFN with raw inputs	15
principal components of IR	ethanol in latex suspension	X	6:3:1[d]	beat PCA, least squares	20
principal components of IR	fat content of meat	X	6:6:1[d]	beat PCA, least squares	20
UV circular dichroism spectrum	proportions of secondary structural types in protein	X	83:45:5	surpassed previous methods on 3 classes, similar on other 2	e
X-ray fluorescence lines	Cr, Fe, Ni content	X	3:3:3:3	beat singular value decomposition	f
ICP atomic emission spectrum	As, Cd concentrations	X	42:2	slightly worse than linear regression	87
	Classification of Spectra of Unknown Compounds				
IR spectrum	aromatic substitution pattern	C	93:31:15:10	≥human experts	86
IR spectrum	functional groups, substructures	B	284:33:1	good categorization	g
IR spectrum	C-O functional groups	B	250:18:6	86.9% right, 6.2% wrong, 6.9% uncertain	96
IR spectrum	above, C-O bond order(s)	B	250:18:9	90.3% right, 3.0% wrong, 6.7% uncertain	96
IR spectrum	functional groups, substructures	B	256:128	identified 53.3% with 91.5% accuracy	85
IR spectrum	functional groups, substructures	B	256:34:36	9 groups > 90% right, 11 > 75%	97
IR, ^{13}C NMR, molecular formula	functional groups, substructures	B	512:48:86	average A50[h] 82%, better than STIRS	53
MS peak intensities	adulteration of olive oil	B	150:8:1	100% correct on new samples	i
features of MS	presence of steroid structure	B	8:100:1	similar to discriminant analysis	j
MS: cations, neutral losses, etc.	functional groups, substructures	B	various	beats STIRS, worse than DENDRAL[k]	34
	Transformation of Spectra into Related Functions				
masses, positions, and momenta of atoms in tetraatomic molecule	internal mode energies	X	28:38:12:6	error averages 1–12%	91
IR spectrum of polyethylene	potential energy function	X	426:7:18	error <3.9% extrapolating, <1% interpolating	92
time, excitation, temperature	CH mode energy in polyethylene	X	3:21:3:1	test error 2.3%	93
small part of spectrum, noise data	probability of peak	X	14:2:1	rms error 0.12	24
small part of 2D NMR spectrum	shape of cross peak	X	16·16:16:1	gives right answer on real data	89
excitation pattern Fourier coefficients	coefficients of pulse shape	X	100:200:15	experiment using FFN-generated pulse shape works correctly	18
polymer heat capacity 110-350 K	heat capacity 10-100 K	X	25:15:10	results well within usual experimental error	90

[a] Explanations for the headings are given in the footnotes to Table I. [b] PCA = principal components analysis (explained in text). [c] Long, J. R.; Gregoriou, V. G.; Gemperline, P. J. *Anal. Chem.* **1990**, *62*, 1791–1797. [d] This FFN includes direct connections between input units and output units. [e] Böhm, G.; Muhr, R.; Jaenicke, R. *Protein Eng.* **1992**, *5*, 191–195. [f] Bos, M.; Weber, H. T. *Anal. Chim. Acta* **1991**, *247*, 97–105. [g] Ricard, D.; Cachet, C.; Cabrol-Bass, D.; Forrest, T. P. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 202–210. [h] A50 is the percentage of correct positive classifications when the threshold is set to the median network output of the true positive examples. [i] Goodacre, R.; Kell, D. B.; Bianchi, G. *Nature* **1992**, *359*, 594. [j] Lohninger, H.; Stancl, F. *Fresenius J. Anal. Chem.* **1992**, *344*, 186–189. [k] STIRS searches for similar examples in an MS database (ref 82); DENDRAL applies rules developed by a human expert (ref 83).

shape from desired excitation patterns reproduced results found by simulated annealing. One new pulse shape produced the desired NMR spectrum when used in an actual experiment. Although the training time was long, the network predicted new examples much faster than simulated annealing—enough to make it feasible to use this technique to design pulse shapes *during* an analysis.

3. Classification

The interpretation of IR spectra is a natural application of FFNs: functional groups in a molecule absorb with varying wavelengths and strengths depending on complex interactions with other parts of the molecule. Several groups have developed networks that identify functional groups present in a compound from its IR spectrum.[85,86,96–98] The ability of an FFN to recognize a group depended on both its distinctiveness in the IR spectrum and the frequency with which it occurred in the training set: a network recognized carbon–halogen bonds well (even though the IR band of this group varies in intensity and position) unless the proportion of such compounds in the training set decreased below one-fifth.[97] Munk, Madison, and Robb extended their method by including not only IR data but ^{13}C NMR peaks (separated by multiplicity) and the molecular formula.[53] The combined-input network outperformed STIRS, a "nearest-neighbor" program that classifies spectra by their similarity to examples in a library.[99]

Curry and Rumelhart developed a neural network that classified compounds by substructure from their mass spectra.[34] As a generator of structures from spectra, this network paralleled STIRS and DENDRAL, an expert system that classifies compounds from mass spectra by applying rules provided to it by a human expert.[100] The accuracy of these networks roughly equaled that of the STIRS program, but the network operated 100 times faster in analyzing new spectra (after being trained). The results did not measure up to those of the DENDRAL program, but that program covered only a smaller set of compounds for which chemists could explicitly encode fragmentation pathways. The authors have given careful thought to every stage of the process; we cite their work as one model for future applications of FFN methodology.

The connectionist approach has been applied to the interpretation of mass spectra since the invention of the perceptron,[101] but this application still presents special problems to the FFN method. For example, features representing characteristic losses depend on the distance between a pair of peaks, rather than the absolute position of the pair. Curry and Rumelhart used five different types of input to capture features dependent on relative positions of peaks as well as those with absolute positions. Peak heights of ion masses from 40 to 219 identified ionic fragments with a recognizable mass. Peak heights corresponding to neutral losses of mass 0–179 from M^+ represented neutral fragments of characteristic mass. Autocorrelation sums, series sums, and constant sums reflected regular patterns in the peaks independently of their position relative to the origin or M^+. The authors could do such detailed analysis in part because they had enough training examples (over 30 000) to allow generalization with many inputs.

The outputs of the network corresponded to the presence of specific substructures. The authors found that, under some training regimes, intermediate values of the output corresponded to actual conditional probabilities. For major substructures, they trained a separate network to identify substructures only on the basis of examples in that particular class in order to help the network improve the quality of prediction on more difficult substructures.

The authors modified the error function in three ways. The network ignored errors smaller than a lower threshold value during training to avoid overfitting. Because of the danger that an example might be a mistake in the database or otherwise misleading, the authors also placed a ceiling on derivatives of the error function so that outliers (examples with large errors) would not affect the training process too much. In some experiments, the error function was divided by the frequency of the class of the example to allow all classes, however rare, to affect training equally. (Despite such compensation, the difficulty of abstracting from only a few examples generally makes recognizing rare classes more difficult.[57])

Besides the changes in the error function, the authors made three other modifications to the training method. First, they added Gaussian noise to the training inputs to prevent overfitting. Second, training of the network stopped when the error on a cross-validation test set reached a minimum. Third, by a process the authors call "automatic weight elimination", small weights shrank when the error fell below a given threshold until the error rose back to the threshold. This process gradually reduced small weights with insignificant effects to zero.

Cross-validation procedures assessed the architecture of the network. The authors found that the network learned most efficiently when the output units shared the hidden units instead of dividing them: a fully-connected layered network required fewer hidden units to reach the same accuracy as a network in which each hidden unit was connected to only one output. They also found that the accuracy of the network increased with the number of hidden units as long as error on a test set was used to end training. As the number of hidden units increased, the minimum error on the test set decreased, the number of training cycles necessary to train the network decreased, and the quickness of the network to overfit (i.e., to increase error on the test set after reaching a minimum) increased. Without that evidence, it would seem equally plausible to assume that a large network could fit a given set of training data in more ways, reducing the likelihood that a trained network would predict well. Automatic weight elimination may have contributed to the predictivity of their larger networks.

D. Applications in Sensor Arrays (Unordered Homogeneous Data)

Three factors limit the utility of chemical sensors to characterize samples: sensors are not perfectly selective; a sensor may respond in a nonlinear manner to a complex mixture; existing analytical techniques may not extract all the information in the data. The data consist of the outputs of different sensors. An FFN transforms the responses from these sensors—a position in "selectivity space"—into the recognizable output of identity or concentration of the sample. One can consider this interpretation of data from sensor arrays as a kind of spectroscopy. FFNs have recognized samples and quantified them on the basis of the data from sensors (Table IV).

The sense of smell recognizes molecules in the gas phase. Sensor technology has mimicked that sense in the analysis of fragrances.[102,103] Quartz resonators attached to selectively adsorbent membranes sense an amount of material adsorbed as a change in frequency. A Japanese group trained networks to distinguish between similar whiskies, between alcoholic food products, between perfumes, and between fruit flavors and claimed that they outperformed both linear discrim-

Table IV. Applications of FFNs to Problems in Interpreting the Output of Sensor Arrays[a]

input	output	type	architecture	results	ref
	Gases				
SnO_2 gas sensors, temperature gradient	3 compounds, binary mixtures	C	17:8:6	85% correct	111
difference between adjacent sensors	3 compounds, binary mixtures	C	16:8:6	96% correct	111
all above inputs	3 compounds, binary mixtures	C	33:8:6	92% correct	111
lipids on quartz resonator sensors	3 odorants, 4 alcohols	C	6:7:7	imperfect recognition	108
tin oxide sensors	hexane, acetone, aqueous NH_3	C	?	correct recognition	*b*
tin oxide sensors	2-methyl-1-butene, benzene, gasoline	C	?	correct recognition	*b*
tin oxide sensors	3 beers	C	12:4:3	correct recognition	110
tin oxide sensors, fractional conductance change	3 beers	C	12:7:3	correct recognition	109
tin oxide sensors, fractional conductance change	5 alcohols	C	12:7:5	correct recognition	109
membrane/quartz resonator sensors	liquors	C	6:8:11	73% correct vs 64% linear discrimination	107
same sensors, EtOH subtracted	liquors	C	6:8:11	88% correct vs 83% linear discrimination	107
membrane/quartz resonator sensors	whiskies	C	3:3:5	88% correct vs 83% linear discrimination	106
lipid bilayers on quartz resonators	whiskies	C	8:7:5	94% correct	105
lipid bilayers on quartz resonators	perfumes	C	8:7:5	100% correct	104
lipid bilayers on quartz resonators	fruit flavors	C	8:7:5	100% correct	104
lipid bilayers on quartz resonators	adulterated orange flavors	C	8:7:5	threshold for detection of adulterant same as that of human nose	104
metal oxide sensors (MOSFET)	H_2, NH_3, C_2H_4, EtOH concentrations	X	6:4:4	FFN better than partial least squares	112
metal oxide sensors (MOSFET)	H_2, acetone concentrations	X	6:4:2	FFN better than partial least squares	112
	Ions				
ion-selective electrodes	Ca, Cu ion concentrations	X	4:5:2	reasonable predictions	32
ion-selective electrodes	K, Ca, NO_3, Cl ion concentrations	X	5:7:4	reasonable predictions	32

[a] Explanations for the headings are given in the footnotes to Table I. [b] Nakamoto, T.; Takagi, H.; Utsumi, S.; Moriizumi, T. *Sens. Actuators B* **1992**, *8*, 181–186.

ination and human noses.[104–107] Similar work by Chang et al. had less success.[108] A group at Warwick University has worked on a method to discriminate between beers[109] and to monitor the odor of ale in brewing with tin oxide sensors.[110]

Tin oxide sensors on a temperature gradient have been used to recognize simple organic molecules and their binary mixtures. The best recognition resulted from presenting the derivative of sensor response with respect to temperature as input to the network.[111] Too much information, as well as too little, can impede the ability of a network to learn: prediction suffered when both raw inputs and derivatives were given to a network. Ion-selective electrodes[32] and gas sensors[112] have also formed the basis for accurate quantitation.

E. Applications in QSAR (Various Data)

Quantitative structure–activity relationships pose a special problem for FFNs. Unlike the subjects of other applications of FFNs in chemistry, molecules lack an obvious numerical model. Ignorance of the physical basis of the phenomenon with which molecular properties are being correlated exacerbates the problem of representation. In many cases, researchers have restricted their set of examples to one parent compound and its substituted analogues, allowing them to limit the description of the molecule to simple physical properties of the substituents. For a general representation, molecular graphs, the familiar line drawings used in organic chemistry, can be translated into a list of numbers. In practice, that translation usually includes simple chemical information. The complexity of this field makes QSAR an ideal proving ground for FFNs (Table V), the most flexible generalizing method available.

1. QSAR through Representations of Connectivity

Representing a set of connections between atoms as inputs to an FFN can be accomplished in one of three ways. First, one can use input values to represent both the atoms of the molecule and the bonds between them, as a connection table or as a matrix of the bond orders between every pair of atoms.[113] In such methods, the assignments of particular atoms to particular inputs does not matter, at least in theory. Second, one can assume a given set of connections between the atoms, forming a template. Each input unit corresponds to an atom in a particular position in the molecular template. If some bonds in the molecule under consideration are absent in the template, the inputs cannot represent the molecule properly. Third, one can use inputs that represent the presence or number of particular substructures in the molecule. These three methods may apply either to the entire molecule or only to a particular substituent.

Elrod, Maggiora, and Trenary used simplified connectivity matrices to predict the products of addition reactions.[114] In this representation, the input consisted of an $n{\cdot}n$ matrix, where n is the number of atoms in the core reactive group and its immediately adjoining atoms. Each row and column corresponded to one atom; each entry corresponded to the bond order between the atom of the row and the atom of the column. Atomic numbers of the atoms filled the diagonal entries. In earlier works,

Table V. Applications of FFNs to Problems in Quantitative Structure–Activity Relationships[a]

input	output	type	N[b]	architecture	results	ref
			Connectivity Tables			
atomic numbers, connections	meta yield in aromatic substitution	X	32	5-5:5:2	10/13, same as chemists, better than CAMEO[c]	115
simplified Dugundji–Ugi matrix	matrix after Markovnikov addition	X	18	36:8:8:28	high but suspect prediction	114
simplified Dugundji–Ugi matrix	matrix after Saytzeff elimination	X	72	66:24:24:55	high but suspect prediction	114
simplified Dugundji–Ugi matrix	matrix after Diels–Alder reaction	X	30	120:36:36:105	high but suspect prediction	114
			Molecular Template			
arbitrary coding for carbons	retrosynthetic reactions	C	32	8:4:4	80% "reasonable"; only 3 examples shown	117
presence of carbon at position	^{13}C NMR peak for secondary carbon	X	40	Figure 6	incremental method better	44
atomic properties for α, β atoms	aromatic substitution, meta yield	X	21	Figure 6	22.1% rms error	43
			Descriptors from Molecular Graph			
sums of atomic properties for α, β atoms	aromatic substitution, meta yield	X	21	9:4:4:1	11.5% rms error	43
sums of atomic properties for α, β atoms	^{13}C NMR peaks in monosubstituted benzene	X	44	11:6:4; feedback	good prediction for ipso, poor prediction for other atoms	39
sums of atomic properties for α, β atoms	^{13}C NMR peaks in monosubstituted benzene	X	44	11:6:4; recursion	good prediction	40
sums of atomic properties for α, β atoms	inductive and resonance substituent effects	X	33	14:8:2	$r_I = 0.077$, $r_R = 0.108$	118
various ketosteroid atom descriptors	^{13}C NMR spectrum	X	391	13:40:116	rms error 1.0 ppm, MLR 1.8[d]	16
various ketosteroid atom descriptors	^{13}C NMR peak	X	391	13:40:1	rms error 1.1 ppm	16
various ketosteroid atom descriptors (atoms far from C=O)	^{13}C NMR spectrum	X	190	9:41:116	rms error 0.36 ppm	16
various cyclic ketone descriptors	^{13}C NMR spectrum	X	850	11:55:116	rms error 2.9 ppm	16
various cyclic ketone descriptors	^{13}C NMR predictive model	C	3348	40:20:75	similar to nearest-neighbor	119
			Substituent Descriptors			
various	taste (sweet, bitter, none)	C	97	7:6:3:3	78%, $r_s = 0.68$, $r_b = 0.49$, $r_n = 0.47$[e]	123
various	musk odor	X,B	130	24:5:1, 24:7:3:1	slightly worse than linear discrimination	121
various for α, β atoms of aromatic substituents	musk odor	B	70	5-6:6:3:1	77%, discriminant analysis 81%	120
^{13}C NMR shifts in norbornanes	endo vs exo substitution	B	25	7:14:4	12/13 vs 11/13 cluster or linear learning	*f*
mitomycin descriptors	anticarcinogenic activity	C	11	6:12:5	3/5 predicted	*g*
arylacryloylpiperazine descriptors	antihypertensive activity	C	22	7:14:4	6/8 predicted; ALS 62–76%[h]	*g*
carboquinone descriptors	anticarcinogenic activity	X	36	7:12:1	predictions as good as training results	*i*
benzodiazepine descriptors	tranquilizer activity	X	56	13:26:1	prediction worse than MLR	*i*
mitomycin descriptors	anticarcinogenic activity	C	15	5:10:3	worse than other methods	*j*
arylacryloylpiperazine descriptors	antihypertensive activity	C	28	6:10:4	worse than other methods	*j*
molecular descriptors	DHFR inhibitory activity	X	100	5:3:1	better than MLR with indicators	51
molecular descriptors	DHFR inhibitory activity	X	34	4:6:1	better than regression analysis	*k*
substituent descriptors, concentration	hypotensive activity	X	24	6:3:1	one outlier, like ALS	*l*
molecular descriptors	mutagenicity	X	147	4:4:1	similar to linear regression	*m*
substituent, molecular descriptors	carcinogenicity	B	46	5–11:7:2	adequate training results	*n*
identities of cation and anion	solubility	C	538	55:2:3	reasonably good prediction	29
various	logarithm of solubility	X	331	17:18:1	rms error 0.43 vs 0.36 for MLR	*o*
identities of dihydropyridine substituents	ΔH of oxidation	X	71	22:4:1	rms error 1.8 kcal/mol	*p*
molecular electrostatic potentials	pK_a of imidazole	X	29	144:9:1	rms error 0.8	*q*
atomic properties	acidity of mixed oxides[r]	B	38	12:10:10:8	better than previous correlation	*s*
functional group types	biodegradability	X	18	8:8:1	better than linear model	*t*
atom types	biodegradability	C	79	?	no prediction, poor training	*u*
functional group types	biodegradability	C	51	?	no prediction, good training	*u*

[a] Explanations for the headings are given in the footnotes to Table I. [b] Number of training examples. [c] CAMEO is an expert system that applies rules on the basis of chemical mechanisms (Gushurst, A. J.; Jorgensen, W. L. *J. Org. Chem.* **1988**, *53*, 3397–3408). [d] MLR = multiple linear regression. [e] r_s, r_b, and r_n are the correlation coefficients for the categories sweet, bitter, and tasteless, respectively. [f] Aoyama, T.; Suzuki, Y.; Ichikawa, H. *Chem. Pharm. Bull.* **1989**, *37*, 2558–2560. [g] Aoyama, T.; Suzuki, Y.; Ichikawa, H. *J. Med. Chem.* **1990**, *33*, 905–908. [h] ALS = adaptive least squares (Moriguchi, I.; Komatsu, K.; Matsushita, Y. *J. Med. Chem.* **1973**, *23*, 20). [i] Aoyama, T.; Suzuki, Y.; Ichikawa, H. *J. Med. Chem.* **1990**, *33*, 2583–2590. [j] Liu, Q.; Hirono, S.; Moriguchi, I. *Quant. Struct.-Act. Relat.* **1992**, *11*, 318–324. [k] So, S.-S.; Richards, W. G. *J. Med. Chem.* **1992**, *35*, 3201–3207. [l] Wiese, M. *Quant. Struct.-Act. Relat.* **1991**, *10*, 369–371. [m] Ghoshal, N.; Mukhopadhyay, S. N.; Ghoshal, T. K.; Achari, B. *Bioorg. Med. Chem. Lett.* **1993**, *3*, 329–332. [n] Adler, B.; Ammon, K.; Dobers, S.; Winterstein, M.; Ziesmer, H. *Chem. Tech.* **1992**, *44* (*11–12*), 363–367. [o] Bodor, N.; Harget, A.; Huang, M.-J. *J. Am. Chem. Soc.* **1991**, *113*, 9480–9483. [p] Brewster, M. E.; Huang, M.-J.; Harget, A.; Bodor, N. *Tetrahedron* **1992**, *48*, 3463–3472. [q] Broughton, H. B.; Green, S. M.; Rzepa, H. S. *J. Chem. Soc., Chem. Commun.* **1992**, 1178–1180. [r] The acidity is represented as a sequence of binary variables representing particular pH values. [s] Kito, S.; Hattori, T.; Murakami, Y. *Ind. Eng. Chem. Res.* **1992**, *31*, 979–981. [t] Tabak, H. H.; Govind, R. *Environ. Toxicol. Chem.* **1993**, *12*, 251–260. [u] Zitko, V. *Chemosphere* **1991**, *23*, 305–312.

they used connection tables with serial numbers to predict the degree of meta direction in electrophilic aromatic substitution.[115] Each atom had inputs corresponding to its atomic number and the serial numbers of the atoms to which it was connected. In both methods, they also used the template model: the authors aligned the molecules so that corresponding atoms in different examples were assigned to the same input, ensuring that the arrangement of chemical bonds would remain somewhat consistent.[116]

We believe that successful prediction in their work arose from the use of a molecular template, not through connectivity. In a connection table, some entries for a particular atom refer to other atoms in the table. Such a representation comes naturally to digital computers, in which many variables indicate the locations of other variables. In contrast, the units of an FFN add input variables together (eq 1). Although a network can make one input point to another (that is, an input indicating a bond to particular atom enhances the effects of the inputs corresponding to that other atom), such an implementation requires one hidden unit for each property for each connection of interest. The large number of hidden units resulting from such a representation would require many more training examples than the number provided in their work.

Luce and Govind used a template for atoms from ϵ to β' in a molecule containing a C=O or C=NR group.[117] They represented each atom by a number assigned to a particular substructure. Their assignment of numbers to moieties has no clear meaning: benzyl and nitrile groups have very similar numbers despite the difference in their properties. Because the authors reduced the molecules to a list of moieties, they could have used a sparsely-coded matrix as input. Such a representation would have rendered their arbitrary encoding unnecessary.

Kvasnička used templates for alkanes[44] and substituents on aromatic rings[43] in FFNs for predicting the positions of ^{13}C NMR peaks and yield of meta products in electrophilic aromatic substitution, respectively. These networks had connection patterns that paralleled the bonds in the templates rather than fully connected layers (Figure 6). As intriguing as this method is, predictions using it did not perform as well as an incremental method (in ^{13}C NMR) and a simpler layered FNN (in yields of substitution products). In later work, the inputs from the template were combined into counts of particular substructures. Kvasnička and coworkers used standard FFNs to predict aromatic substituent effects[118] and FFNs with extra recursive connections to predict ^{13}C NMR peaks in monosubstituted benzenes.[39,40]

Anker and Jurs applied FFNs to the prediction of the position of ^{13}C NMR peaks of ketosteroids from various descriptors derived from the molecular graph and physical properties of the molecule.[16] As output, they used a series of units representing the possible presence of the peak at 116 points in the spectrum spaced 0.5 ppm apart, rather than a single value for the chemical shift. This scheme provided not only more accurate results than a network predicting the chemical shift directly, but also a confidence value, generated from relative magnitudes of the outputs. Their method achieved greater accuracy on a subset of atoms far from

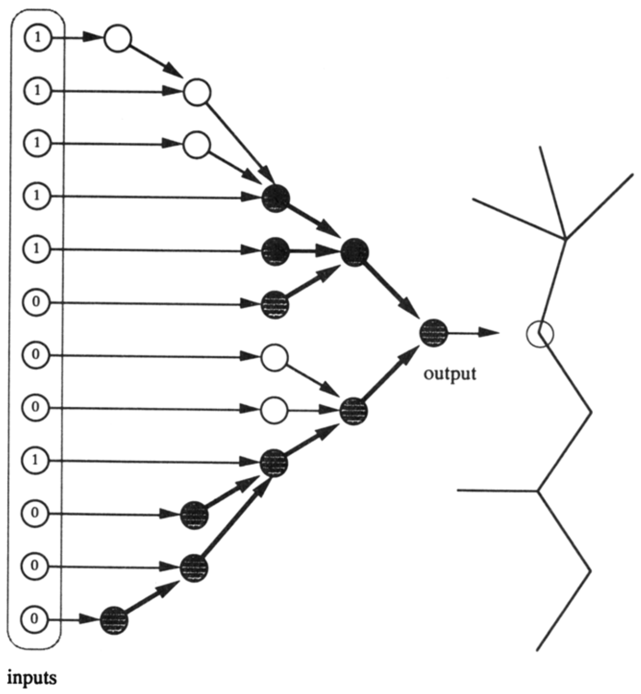


Figure 6. The architecture of the neural network designed by Kvasnička (ref 44) reproduces part of the possible skeleton of an alkane. The values of the external units correspond to the presence or absence of a carbon at that position. The units representing carbon atoms are shaded. The pattern shown corresponds to 2,2,5-trimethylheptane; the output represents the ^{13}C chemical shift of the circled carbon atom.

the carbonyl group in the steroids than on the complete set. It also predicted spectra for other cyclic ketones when trained on a general set of them, with somewhat less accuracy than the network achieved on ketosteroids alone. Ball and Jurs trained a network from similar inputs to select the best model to use to predict ^{13}C NMR peaks.[119]

2. QSAR from Substituent Properties

A variety of properties have been used to describe substituents or characteristics of a molecule. Binary values indicate the presence of groups or conditions. Topological indices and descriptors express molecular connectivity. Geometrical properties encompass distances, angles, and similar values. More physical properties include charge, electronegativity, and substituent constants; hydrophobicity terms; and van der Waals repulsions or molar volumes.

Because no good physical models for smell and taste exist, one might imagine that FFNs would surpass rival methods in predicting them from structure. So far, they have not, although few examples exist. Autocorrelation and linear discriminants have proved more effective than FFNs in predicting musk odor.[120,121] FFNs predicted sweetness in L-aspartyl dipeptides from molecular descriptors better than SIMCA, a form of principal component analysis used for classification,[122] but bitterness worse.[123] FFNs have had relative success in the prediction of drug activity and simple chemical properties, although some applications did not yield results as good as the best existing methods do (see Table V).

3. Summary

An FFN can often, but not always, improve on existing methods of making predictions from a given data set if the size of the data set permits generalization. Unfortunately, the number of examples available for typical problems in QSAR is small: only four of the applications included more than 200 examples in the training set. In virtually every case, the number of weights exceeded the number of training examples. Smaller networks might have sufficed, but few researchers based their selection of architecture on performance. Many of these results might improve with the application of current best practice, especially cross-validation and selection of architecture based on predictive ability. Even with these problems, FFNs generally compare favorably with existing methods.

IV. Discussion

A. Review of Applications

What is the state of the art of applications of FFN technology to problems in chemistry? The level of accomplishment depends on finding a good numerical representation for the problem under study.

The idea of sparse coding of substituents occurred naturally in considering biological macromolecules, for which substituents are few and well-defined. The networks succeeded in deriving the effects of particular residues, even though no data explicitly describing properties had been provided to them. Researchers have begun to use FFNs as tools rather than as objects of study. The same method should work for substituents on other molecules, although the network needs many examples of combinations of substituents to learn the effects of each one. When only data describing properties are provided to the network, it is easily possible to omit important properties, or to define them in perverse ways.

In both spectroscopy and analysis of genes, researchers have improved results by preprocessing the obvious representation. In part, these techniques draw on the knowledge that higher-order features, such as codons in DNA or fragmentation patterns in MS, play a role in the underlying chemistry or biochemistry of the phenomenon being studied. Proton NMR spectra contain much information, but FFNs have so far extracted relatively little of it because so much of it depends on higher-order features caused by coupling.

The application of FFNs to QSAR has not yet reached maturity. Almost all the work done so far involves limited problems with few training examples and ad hoc construction of input sets. The work on predicting molecular properties from substituent data is adequate and informative, but in most cases, not a great improvement over existing methods. The scarcity of training examples and the absence of efficient methods of representing important characteristics such as molecular shape hinder these applications more than any other factors.

The best method of representing molecular structures for FFNs is not yet clear. Although the results have not surpassed other methods, describing a molecule in terms of its connections holds promise as a way to represent molecules in a completely general way. The bond-graph architecture of Kvasnička's network[43,44] is an imaginative approach to providing implicit connectivity information to a network: by cutting connections between portions of the network corresponding to distant atoms, one can reduce the size of the net and emphasize important relationships.

Elrod, Maggiora, and Trenary attempted to use connection tables to represent connections explicitly.[114,115] The consistent numbering of the atoms and the scarcity of training examples undermined their application, but the principle remains sound. Because modeling the "switching" or "pointing" behavior of the units requires many hidden units, an application of this method will need a much larger training set than those used so far. Providing multiple representations of the same molecule (in the case of a Dugundji–Ugi matrix, switching the order of the rows and columns) can help.

B. Review of Methodology

Many of the earlier papers cited in this review have not used the best available methods, but the sophistication of applications of FFNs has increased rapidly with time. The simple, general FFN suffices for most purposes. To prevent overtraining and to choose an architecture, many researchers have used simple rules of thumb. We do not recommend using those rules. Although trimming of unnecessary weights and units can reduce overfitting in overtrained or excessively large networks, we believe that only performance on test data (often reserved by cross-validation) can define the quality of a trained network. Using performance as the criterion for training, one can optimize the predictivity of a network, rather than its fit to training data. Any feature of the network can be optimized by this method: architecture, training time, or training parameters. One can even assess the reproducibility of the results by taking several different divisions between training and cross-validation test data, as one does in bootstrapping statistics.[20,47]

C. Conclusion

Enough impressive results have been obtained with FFNs to suggest that they should be a useful technique in chemistry, particularly when complex, poorly understood, but well-defined data (such as spectra, sensor readings, or biological sequences) are involved. Despite the hypothetical advantages of FFNs for problems involving hard-to-define properties, few of the existing applications have made full use of them, or surpassed the accuracy of other methods. Combining the best examples of thoughtful, tested uses for this technique can improve the overall effectiveness of FFNs in all fields of chemistry that need computational tools.

Acknowledgments. This work was supported by the NIH through grant no. GM39589 and by Firmenich SA.

V. References

(1) We have chosen the abbreviation FFN to refer to the described methods in order to distinguish them from other paradigms that also fit in the category of "artificial neural networks" (ANNs), especially unsupervised techniques such as Kohonen maps which have also found some use in problems in chemistry. Unsupervised networks are trained to find patterns in sets of values; supervised networks are trained to generate correct outputs from inputs. FFNs are sometimes called "back-propagation neural networks", after the method usually used to train the network.

(2) The difference is one of degree, not kind: physical principles originate in empirical data, and the definition of "internal consistency" is often based on physically reasonable criteria.
(3) Woodruff, H. B.; Lowry, S. R.; Isenhour, T. L. *Anal. Chem.* **1974**, *46*, 2150-2154.
(4) Quinlan, J. *Mach. Learn.* **1986**, *1*, 81-106.
(5) Minsky, M.; Papert, S. *Perceptrons*; MIT Press: Cambridge, MA, 1969.
(6) Hornik, K.; Stinchcombe, M.; White, H. *Neural Networks* **1989**, *2*, 359-366.
(7) Rumelhart, D. E., McClelland, J. L., Eds. *Parallel Distributed Processing*; Cambridge, MA: MIT Press, 1986. McClelland, J. L.; Rumelhart, D. E. *Explorations in Parallel Distributed Processing*; MIT Press: Cambridge, MA, 1988.
(8) Wasserman, P. D. *Neural Computing: Theory and Practice*; Van Nostrand Reinhold: New York, 1989.
(9) Hertz, J.; Krogh, A.; Palmer, R. G. *Introduction to the Theory of Neural Computation*; Addison-Wesley: Reading, MA, 1991.
(10) Caudill, M.; Butler, C. *Naturally Intelligent Systems*; MIT Press: Cambridge, MA, 1990.
(11) Beale, R.; Jackson, T. *Neural Computing*; IDP Publishing: New York, 1990.
(12) Zornetzer, S. F., Davis, J. L., Lan, C., Eds. *An Introduction to Neural and Electronic Networks*; Academic Press: San Diego, 1990.
(13) Müller, B.; Reinhardt, J. *Neural Networks: An Introduction*; Springer-Verlag: Berlin, 1990.
(14) Eberhart, R. C., Dobbins, R. W., Eds. *Neural Network PC Tools: A Practical Guide*; Academic Press: San Diego, 1990.
(15) Gemperline, P. J.; Long, J. R.; Gregoriou, V. G. *Anal. Chem.* **1991**, *63*, 2313-2323.
(16) Anker, L. S.; Jurs, P. C. *Anal. Chem.* **1992**, *64*, 1157-1164.
(17) McAvoy, T. J.; Su, H. T.; Wang, N. S.; He, M.; Horvath, J.; Semerjian, H. *Biotechnol. Bioeng.* **1992**, *40*, 53-62.
(18) Gezelter, J. D.; Freeman, R. *J. Magn. Reson.* **1990**, *90*, 397-404.
(19) Bohr, H.; Bohr, J.; Brunak, S.; Cotterill, R. M. J.; Fredholm, H.; Lautrup, B.; Petersen, S. B. *FEBS Lett.* **1990**, *261*, 43-46.
(20) Borggaard, C.; Thodberg, H. H. *Anal. Chem.* **1992**, *64*, 545-551.
(21) Uberbacher, E. C.; Mural, R. J. *Proc. Natl. Acad. Sci. U.S.A.* **1991**, *88*, 11261-11265.
(22) Viswanadhan, V. N.; Denckla, B.; Weinstein, J. N. *Biochemistry* **1991**, *30*, 11164-11172.
(23) Zhang, X.; Mesirov, J. P.; Waltz, D. L. *J. Mol. Biol.* **1992**, *225*, 1049-1063.
(24) Wythoff, B. J.; Levine, S. P.; Tomellini, S. A *Anal. Chem.* **1990**, *62*, 2702-2709.
(25) Lippmann, R. P. *IEEE ASSP Mag.* **1987**, *4* (2), 4-22.
(26) In early work, the bias term is often missing or given a fixed value. This limitation reduces the flexibility of each unit and the ability of the network to generalize.
(27) Darken, C.; Moody, J. *Neural Inf. Process. Sys.* **1991**, *3*, 832-838.
(28) Demeler, B.; Zhou, G. *Nucleic Acids Res.* **1991**, *19*, 1593-1599.
(29) Burns, J. A.; Whitesides, G. M., *J. Phys. Chem.* Submitted for publication.
(30) Silva, F. M.; Almeida, L. B. *Adv. Neural Comput.* **1990**, 151-158.
(31) Kruschke, J. K.; Movellan, J. R. *IEEE Trans. Sys. Man Cybern.* **1991**, *21*, 273-280.
(32) Bos, M.; Bos, A.; Van der Linden, W. E. *Anal. Chim. Acta* **1990**, *233*, 31-39.
(33) Aoyama, T.; Ichikawa, H. *Chem. Pharm. Bull.* **1991**, *39*, 372-378; (erratum), 1649.
(34) Curry, B.; Rumelhart, D. E. *Tetrahedron Comput. Methodol.* **1990**, *3*, 213-237.
(35) Qian, N.; Sejnowski, T. J. *J. Mol. Biol.* **1988**, *202*, 865-884.
(36) Makram-Ebeid, S.; Sirot, J. A.; Viala, J. R. *IEEE/INNS Int. Joint Conf. Neural Networks* **1989**, 373-380.
(37) Muskal, S. M.; Kim, S.-H. *J. Mol. Biol.* **1992**, *225*, 713-727.
(38) Owens, A. J.; Filkins, D. L. *IEEE/INNS Int. Joint Conf. Neural Networks* **1989**, 381.
(39) Kvasnička, V.; Sklenák, Š.; Pospíchal, J. *J. Mol. Struct. (Theochem)* **1992**, *277*, 87-107.
(40) Kvasnička, V.; Sklenák, Š.; Pospíchal, J. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 742-747.
(41) Polak, E. *Computational Methods in Optimization*; Academic Press: New York, 1971.
(42) Fletcher, R.; Powell, M. J. D. *Comput. J.* **1963**, *6*, 163.
(43) Kvasnička, V.; Pospíchal, J. *J. Mol. Struct. (Theochem)* **1991**, *235*, 227-242.
(44) Kvasnička, V. *J. Math. Chem.* **1991**, *6*, 63-76.
(45) Goldberg, D. *Genetic Algorithms in Search Optimisation and Machine Learning*; Addison-Wesley: Reading, MA, 1989.
(46) Kirkpatrick, S.; Gelatt, C. D., Jr.; Vecchi, M. P. *Science* **1983**, *220*, 671-680.
(47) Peterson, I. *Sci. News* **1991**, *140*, 56-58.
(48) Chester, D. L. *IEEE Int. Joint Conf. Neural Networks* **1990**, I 265-268.
(49) Lapedes, A. S. *Book of Abstracts*, 203rd ACS Meeting, San Francisco, California, 1992; American Chemical Society, Washington, DC, 1992; Abstract COMP-33.
(50) Lapedes, A.; Barnes, C.; Burkes, C.; Farber, R.; Sirotkin, K. In *Computers and DNA: Santa Fe Institute Studies in the Science of Complexity VII*; Addison-Wesley: Reading, MA, 1989.
(51) Andrea, T. A.; Kalayeh, H. *J. Med. Chem.* **1991**, *34*, 2824-2836.
(52) This rule is implicit in many papers that do not explain their choice of architecture.
(53) Munk, M. E.; Madison, M. S.; Robb, E. W. *Book of Abstracts*, 203rd ACS Meeting, San Francisco, California, 1992; American Chemical Society: Washington, DC, 1992; Abstract COMP-36.
(54) Thomsen, J. U.; Meyer, B. *J. Magn. Reson.* **1989**, *84*, 212-217.
(55) Holley, L. H.; Karplus, M. *Proc. Nat. Acad. Sci. U.S.A.* **1989**, *86*, 152-156.
(56) McGregor, M. J.; Flores, T. P.; Sternberg, M. J. E. *Protein Eng.* **1990**, *3*, 459-460.
(57) Dr. Bo Curry mentioned this problem during a question and answer session at the neural network symposium at the 203rd ACS Meeting, San Francisco, California, on April 9, 1992.
(58) Matthews, B. W. *Biochem. Biophys. Acta* **1975**, *405*, 442-451.
(59) Stolorz, P.; Lapedes, A.; Xia, Y. *J. Mol. Biol.* **1992**, *225*, 363-377.
(60) Biou, V.; Gibrat, J. F.; Levin, J. M.; Robson, B.; Garnier, J. *Protein Eng.* **1988**, *2*, 185-191.
(61) Gonnet, G. H.; Cohen, M. A.; Benner, S. A. *Science* **1992**, *256*, 1443-1445.
(62) Levitt, M. *J. Mol. Biol.* **1983**, *170*, 723-764.
(63) McGregor, M. J.; Flores, T. P.; Sternberg, M. J. E. *Protein Eng.* **1989**, 2, 521-526. Erratum: ref. 56.
(64) Engelbrecht, J.; Knudsen, S.; Brunak, S. *J. Mol. Biol.* **1992**, *227*, 108-113.
(65) Holley, L. H.; Karplus, M. *Methods Enzymol.* **1991**, *202*, 204-224.
(66) Hirst, J. D.; Sternberg, M. J. E. *Biochemistry* **1992**, *31*, 7211-7218.
(67) Bohr, H.; Bohr, J.; Brunak, S.; Cotterill, R. M. J.; Lautrup, B.; Nørskov, L.; Olsen, O. H.; Petersen, S. B. *FEBS Lett.* **1988**, *241*, 223-228.
(68) Kneller, D. G.; Cohen, F. E.; Langridge, R. *J. Mol. Biol.* **1990**, *214*, 171-182.
(69) Chou, P. Y.; Fasman, G. D. *Biochemistry* **1974**, *13*, 211-245.
(70) Hayward, S.; Collins, J. F. *Proteins: Struct., Funct., Genet.* **1992**, *14*, 372-381.
(71) Poliac, M. O.; Wilcox, G. L.; Xin, Y.; Carmeli, T.; Liebman, M. *IEEE Int. Joint Conf. Neural Networks* **1991**, 1323-1329.
(72) Wilcox, G. L.; Poliac, M.; Leibman, M. N. *Tetrahedron Comput. Methodol.* **1990**, *3*, 191-211.
(73) O'Neill, M. C. *Nucleic Acids Res.* **1991**, *19*, 313-318.
(74) O'Neill, M. C. *Nucleic Acids Res.* **1992**, *20*, 3471-3477.
(75) Brunak, S.; Engelbrecht, J.; Knudsen, S. *J. Mol. Biol.* **1991**, *220*, 49-65.
(76) Farber, R.; Lapedes, A.; Sirotkin, K. *J. Mol. Biol.* **1992**, *226*, 471-479.
(77) Mural, R. J.; Einstein, J. R.; Guan, X.; Mann, R. C.; Uberbacher, E. C. *Trends Biotechnol.* **1992**, *10* (1-2), 66-69.
(78) Uberbacher, E. C.; Mann, R. C.; Hand, R. C., Jr.; Mural, R. J. Report ORNL/TM-11741, 1991. Available from NTIS, order no. DE91-007462.
(79) Snyder, E. E.; Stormo, G. D. *Nucleic Acids Res.* **1993**, *21*, 607-613.
(80) Rowley, D. L.; Wolf, R. E., Jr. *J. Bacteriol.* **1991**, *173*, 968-977.
(81) Brunak, S.; Engelbrecht, J.; Knudsen, S. *Nucleic Acids Res.* **1990**, *18*, 4797-4801.
(82) Brunak, S.; Engelbrecht, J.; Knudsen, S. *Nature* **1990**, *343*, 123.
(83) In an analog device, such data do form a continuous function before being recorded as separate data points on a computer.
(84) Meyer, B.; Hansen, T.; Nute, D.; Albersheim, P.; Darvill, A.; York, W.; Sellers, J. *Science* **1991**, *251*, 542-544.
(85) Robb, E. W.; Munk, M. E. *Mikrochim. Acta* **1990**, *1*, 131-155.
(86) Weigel, U.-M.; Herges, R. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 723-731.
(87) Schierle, C.; Otto, M. *Fresenius J. Anal. Chem.* **1992**, *344*, 190-194.
(88) Sellers, J.; York, W.; Albersheim, P.; Darvill, A.; Meyer, B. *Carbohydr. Res.* **1990**, *207*, C1-C5.
(89) Kjaer, M.; Poulsen, F. M. *J. Magn. Reson.* **1991**, *94*, 659-663.
(90) Darsey, J. A.; Noid, D. W.; Wunderlich, B.; Tsoukalas, L. *Makromol. Chem. Rapid Commun.* **1991**, *12*, 325-330.
(91) Sumpter, B. G.; Getino, C.; Noid, D. W. *J. Chem. Phys.* **1992**, *97*, 293-306.
(92) Sumpter, B. G.; Noid, D. W. *Chem. Phys. Lett.* **1992**, *192*, 455-462.
(93) Sumpter, B. G.; Getino, C.; Noid, D. W. *J. Chem. Phys.* **1992**, *96*, 2761-2767.
(94) Geen, H.; Freeman, R. *J. Magn. Reson.* **1990**, *87*, 415-421.
(95) Hardy, C. J.; Bottomley, P. A.; O'Donnell, M.; Roemer, P. *J. Magn. Reson.* **1988**, *77*, 233-250.
(96) Fessenden, R. J.; Györgyi, L. *J. Chem. Soc. Perkin Trans. 2* **1991**, 1755-1762.
(97) Munk, M. E.; Madison, M. S.; Robb, E. W. *Mikrochim. Acta* **1991**, *2*, 505-514.
(98) Meyer, M.; Weigelt, T. *Anal. Chim. Acta* **1992**, *265*, 183-190.
(99) Dayringer, H. E.; Peysna, G. M.; Venkataraghavan, R.; McLafferty, F. W. *Org. Mass Spectrom.* **1976**, *11*, 529-542.
(100) Lindsay, R.; Buchanan, B. G.; Feigenbaum, E. A.; Lederberg, J. *DENDRAL*; McGraw-Hill: New York, 1980.
(101) Kowalski, B. R.; Jurs, P. C.; Isenhour, T. L.; Reilley, C. N. *Anal. Chem.* **1969**, *41*, 695-700.

(102) Persaud, K. C. *Trends Anal. Chem.* **1992**, *11* (2), 61–67.
(103) *The Economist* **1991**, 320 (7726), 97.
(104) Nakamoto, T.; Fukuda, A.; Moriizumi, T. *Sens. Actuators B* **1993**, *10*, 85–90.
(105) Nakamoto, T.; Fukuda, A.; Moriizumi, T.; Asukura, Y. *Sens. Actuators B* **1991**, *3*, 221–226.
(106) Nakamoto, T.; Fukunishi, K.; Moriizumi, T. *Sens. Actuators B* **1990**, *1*, 473–476.
(107) Ema, K.; Yokoyama, M.; Nakamoto, T.; Moriizumi, T. *Sens. Actuators* **1989**, *18*, 291–296.
(108) Chang, S.-M.; Iwasaki, Y.; Suzuki, M.; Tamiya, E.; Karube, I.; Muramatsu, H. *Anal. Chim. Acta* **1991**, *249*, 323–329.
(109) Gardner, J. W.; Hines, E. L.; Tang, H. C. *Sens. Actuators B* **1992**, *9*, 9–15.
(110) Shurmer, H. V.; Gardner, J. W. *Sens. Actuators B* **1992**, *8*, 1–11.
(111) Wong, L.; Takemori, T.; Siegel, M. W. Report CMU-RI-TR-89-20, 1989. Available from NTIS, order no. AD A213359.
(112) Sundgren, H.; Winquist, F.; Lukkari, I.; Lundstroem, I. *Meas. Sci. Technol.* **1991**, *2*, 464–469.
(113) Dugundji, J.; Ugi, I. *Top. Curr. Chem.* **1973**, *39*, 19–64.
(114) Elrod, D. W.; Maggiora, G. M.; Trenary, R. G. *Tetrahedron Comput. Methodol.* **1990**, *3*, 163–174.
(115) Elrod, D. W.; Maggiora, G. M.; Trenary, R. G. *J. Chem Inf. Comput. Sci.* **1990**, *30*, 477–484.
(116) Elrod, D. W., Personal communication.
(117) Luce, H. H.; Govind, R. *Tetrahedron Comput. Methodol.* **1990**, *3*, 143–161.
(118) Kvasnička, V.; Sklenák, Š.; Pospíchal, J. *J. Am. Chem. Soc.* **1993**, *115*, 1495–1500.
(119) Ball, J. W.; Jurs, P. C. *Anal. Chem.* **1993**, *65*, 505–512.
(120) Chastrette, M.; de Saint Laumer, J.-Y. *Eur. J. Med. Chem.* **1992**, *26*, 829–833.
(121) Burns, J. A.; Viola, P.; Whitesides, G. M. *J. Med. Chem.* Submitted for publication.
(122) Wold, S.; Sjostrom, M. In *Chemometrics: Theory and Application*; Kowalski, B. R., Ed.; American Chemical Society: Washington, DC, 1977.
(123) de Saint Laumer, J. Y.; Chastrette, M.; Devillers, J. In *Applied Multivariate Analysis in SAR and Environmental Studies*; Devillers J., Karcher, W., Eds.; 479–521.