Predicting Phosphorus NMR Shifts Using Neural Networks

Geoffrey M. J. West

School of Computing, Staffordshire University, Staffordshire ST5 2JE, England

Received July 5, 1992

This paper outlines a new method, based on neural networks, for predicting NMR shift values. While developed for phosphorus NMR, the method is applicable to any element whose shift is moderated by its α–δ substituents. The substituent–shift relationship is assumed to be a mapping, with the independent structure variables moderating the dependent shift variable. Networks are trained by inputting structural parameters derived from the molecular connection table(s). The associated shift value is the network output. The input parameters are derived in a two stage process. Structures are first translated onto a graph template whose size is fixed for any coordination class of phosphorus. During translation, substituents are ordered on their extended connectivity values. The linearization of the template representation forms a novel substructure code. In the second stage the element symbols in this substructure code are replaced with their subsymbolic or physical values. The results from using electronegativity values as replacements are given. The method can be extended to allow the network itself to derive an optimum value for each element symbol. This offers a means of *automatically* calculating, and applying, additivity-like rule parameters from structures stored in existing NMR topological databases.

INTRODUCTION AND BACKGROUND

NMR shift prediction plays an important part in many computer assisted structural elucidation (CASE) systems such as DENDRAL,[1] ACCESS,[2] and CSEARCH.[3] These systems seek to automate the entire elucidation process, using a strategy called Generate and Test. Here the generate phase produces a set of candidate structures inferred from the spectral data of the unknown. The testing phase evaluates these candidates by comparing their simulated spectra to that of the unknown. A similarity ranking can then be obtained for identifying the most probable candidate. Independent from their use in such systems, highly accurate simulators can also be used as important spectroscopic tools. One potential use is that of automatically screening large spectral databases, which often contain erroneous data. Before describing the approach we have taken, it is worth outlining some of the characteristics of prediction methods and neural networks.

Prediction Methods. While theoretically offering the greatest predictive accuracy, *ab initio*[4] or semi-empirical approaches[5,6] are currently too time consuming for large scale use. In ^{13}C NMR, the remaining empirical methods can be factored into three approaches: molecular mechanics, additivity rule, and substructure code approaches. Each of these relies on the moderating effects of substituents on the focal shift. This influence is significant out to a four bond distance. In ^{13}C these substituent effects were first characterized in the alkanes.[7] Subsequently, they have also been shown to occur in many other structural classes.[8-11] In phosphorus NMR, similar substituent effects have long been suspected.[12]

The most technically superior ^{13}C method uses a molecular mechanics approach to derive sets of predictive linear equations.[13,14] While highly accurate, these equations are also highly restricted, in that they apply only to the structural class from which they were derived. Furthermore, computational complexities currently limit this technique to compounds with restricted conformational freedom, such as cyclic structures.

Additivity rule methods are similar to force field approaches but derive linear equations from topological parameters alone. Recently, more accuracy has been obtained by the inclusion of stereochemical parameters.[15] The additivity methods assign a base shift value to the focus, with increments for the substituent effects. Weighting coefficients account for the effect of substituent distance from the focus. As in the force field approach, the parameter values are highly class specific. Much effort has been directed toward obtaining the parameters for a wide range of classes.[16-19] The success of this approach is reflected by the existence of large additivity rule compilations[20,21] and of programs for their application.[15,16] The continuous change in reference populations, however, often renders the parameters obsolete. As these parameters are currently manually derived, continual rederivation is very costly.

At present, most CASE simulators are based on hierarchical substructure codes such as the HOSE,[22] ACF,[23] and the DARC codes.[24] Simulation requires a reference database of these codes which is usually derived from some topological database. Each individual code has a shift range associated with its ^{13}C focus. Where the reference population is large, such codes can be used with reasonable confidence. Spectra are simulated by factoring a candidate into its substructure codes. Identity and similarity matches are then made with the reference codes. The associated shifts are then combined to give a simulated spectrum. This combining process is usually directed, in that it considers constraints imposed by the complete structure of the candidate. The simulated spectra are then matched with the unknown and ranked. One drawback of these methods is that they neglect any effect of substituent distance from the focus. The complexities involved in deriving any weighting coefficients, however, are such that currently they are not incorporated in any substructure code system. Another limitation of these methods is that they neglect the effects of stereochemistry. Substructure codes incorporating stereochemistry have been available for nearly a decade.[25] Currently, however, most databases still use codes containing only the topology.

Neural Networks. The application of single layer neural networks or perceptrons to chemical problems is not new.[26-29] Perceptrons were much studied in the 1950s and 1960's, and all but abandoned due to their limited capacities. These networks are capable of only representing linearly separable

0095-2338/93/1633-0577$04.00/0

functions,[30] severely restricting their applicability. This limitation can be overcome by including hidden layers to form *multilayer* networks. This was well-known in the 1960s, but, at the time, no procedures for training multilayer networks were available. By the mid-1980s several multilayer training methods had been developed,[31-33] with successes in many applications.[34-36] This has resulted in a renewed interest in neural networks, with multilayer networks now being applied to several chemically related problems.[37-43]

The current literature on neural networks is abundant, and many texts give good introductions to the subject.[44-47] A review of chemically related applications has recently been published.[48]

Properties of Feed-forward Neural Networks. Feed-forward networks are often used as classifiers and have similarities to feature space methods such as nearest neighbor analyses.[49] In both methods, the feature space is partitioned by a series of decision boundaries. In nearest neighbor classifiers, pertinent features are predefined or extracted from the data. These attributes then define the positions of individual data items in the feature space. The structure of the data set itself in this space is then used to define the classification boundaries. In neural networks, these boundaries are defined by the values of the weight vectors. During learning, the weight values are altered to reduce the overall error of the network. This error is the difference between the actual and desired network output(s). This error driven response results in a moderate parameter independence. In many data sets the actual parameters and interdependencies affecting the output are unknown. Networks can often "extract" these relevant parameters, incorporating them into the weight values. However, one drawback of this approach is the difficulty of relating these weight values to the actual parameters of a problem.[50]

Learning in networks is summated over the entire training set, giving an inherent insensitivity to individual data errors. This strongly contrasts with inductive learning techniques normally used in artificial intelligence (AI). Classical AI methods such as ID3,[51] AQ11,[52] or Version Spaces[53] do not have this inherent robustness and require explicit error handling procedures.

Feature space methods have constraints on the confidence of their predictions, and neural networks are no exception. One major constraint is the ratio of the training set size to the network dimensionality. The dimensionality is defined by the number of network weights. There are no rules which explicitly define a value for this ratio, as it varies according to the problem. Several sources, however, do give general guidelines. These range from "several bits/weight",[54] through "6 times the number of weights",[55] to 10 times the number of weights.[56] One treatment of the statistical arguments for such ratios is given in ref 50. We have assumed the minimum value for this ratio to be about 6:1.

While often used as classifiers, the ability of networks to simulate mappings defined by *continuous* functions is well established.[57] The theoretical proof of this has recently been extended to cover the sigmoid class of transfer functions.[58]

Spectroscopic Applications of Networks. Network techniques have now been applied to both mass spectroscopy[59] and infrared classifications.[60,61] A sliding window input technique has been applied for predicting ^{13}C NMR shifts of alkanes.[62] Here the data set is based on one previously used for structure–shift rule induction.[63] The structural parameters input to the network consist of the coordination states of substituents, encoded in binary input vectors. The network is trained on 59 alkanes (C_5 to C_9), covering a shift range of 50 ppm, and tested on 24 C_9 compounds. Comparing the network and symbolic predictions, the authors report the network method as being more accurate. Treelike network architectures have been applied for predicting shifts of secondary carbons in acyclic alkanes.[64] Here, network topology closely mirrors that of the molecule. The training set consisted of some 40 alkanes, with structures input as binary vectors. Once the network is trained, the difference between the actual and expected output(s) is often below 2 ppm.

Phosphorus NMR. There have been no previous attempts to construct any ^{31}P NMR simulators. This is probably due to the small sizes (<1000 compounds) of previous ^{31}P topological databases. Substantial amounts of ^{31}P data exist, but, until the creation of our database, most of these data were only available in printed form.[65] Relative to carbon, ^{31}P NMR has a wide shift range, spanning some 1000 ppm. This results in a higher information entropy from each *individual* resonating atom with respect to the structure. There are, however, two major factors that increase the complexity of ^{31}P NMR. One is due to the multivalent nature of phosphorus, which commonly exists in valencies of 3 or 5 and coordination numbers from 2 to 6. Thus our database is partitioned into many valency and coordination classes. The other factor concerns the total amount of information obtained from each structure. Relative to ^{13}C NMR, this information is quite low, as the majority of structures contain only a single phosphorus atom. Thus while some ^{13}C NMR databases are much smaller than ours in terms of compound numbers, they contain greater amounts of shift data. As an example, the database of Zupan's CARBON system has only 2356 structures but contains over 30 000 ^{13}C NMR shifts.[66]

Further complications arise due to the IUPAC NMR sign convention of 1976.[67] This standardized the *high* frequency NMR shifts of any element to be reported with a positive sign, as in ^{13}C. Previously, ^{31}P NMR shifts were reported with *low* frequencies positive. In addition, older ^{31}P NMR data were often obtained by using reference compounds other than 85% H_3PO_4, which is now accepted as the standard. Unlike the TMS standard of ^{13}C, the shift from H_3PO_4 occurs in the middle of the ^{31}P shift range.

We built our phosphorus database to serve as a source of examples for a ^{31}P NMR shift prediction system. Our original intention was to base this system on symbolic machine learning methods. Extensive analysis of the database suggested that a prediction method based on neural networks was more appropriate. A full report of the results of these database analyses is beyond the scope of this paper and will be presented elsewhere. A summary of the results pertinent to our choice of a neural network based method is given below. Throughout the rest of this paper the terms α, β, γ, and δ *substituents* will be used with reference to the resonating ^{31}P focus.

Characteristics of the ^{31}P Data Set: Structure–Shift Relationships. The database contains some 15 000 compounds and was built using techniques based on the data structures of Molecular Design Ltd's ChemBase program. The data comes from the ^{31}P Handbook,[65] which contains data on all of phosphorus' valency and coordination classes. Each valency, coordination class in the Handbook covers a diverse range of structural types. For prediction purposes, two compound classes in the database are of a significant size. These are the 4 coordinate, 5 valent and the 3 coordinate, 3 valent classes, which shall be referred to as the 4c5v and 3c3v classes, respectively. Figure 1 shows the distribution of shifts in the

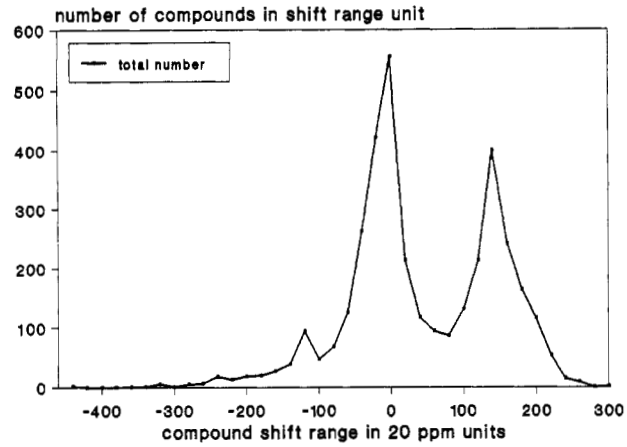


Figure 1. Shift distribution 3c3v class. Compound shifts plotted in 20 ppm ranges.

Table I. ^{31}P Data Set, Characteristics of the Combined Code Created by Concatenating the Handbook Indexing Codes

class	no. of compds in class	no. of types of *combined* code	no. of codes with more than 10 members	av intracode range	largest intracode range
3c3v	3586	1364	44	21.33	178.0
4c5v	6610	3005	60	9.43	113.9

3c3v class. The peak at 0 ppm is from structures with carbon α substituents. The +140 ppm peak is from structures with more electronegative α substituents. The minor peak at −120 ppm is from structures having two hydrogens and 1 group IV element as α substituents. Figure 1 gives strong evidence that the type of α substituent(s) affects the ^{31}P shift value.

Within the Handbook, compounds are ordered on their α and β substituents, and these are grouped separately to form two indexing codes. In these codes the substituents are ordered as in molecular formula, with the hydridization states of substituents also being shown. The concatenation of the two index codes forms a new substructure code, which can be used to measure the structural diversity within the database. Unless stated otherwise, the term *code* hereafter refers to the concatenated indexing codes. The term *code type* refers to a code that is lexicographically unique and corresponds to a distinct structural environment. Here an environment only contains the α and β substituents.

Some code types have more than one example or *member*. These members are structurally indistinguishable when the code is used to measure structural diversity. The number of unique environments, given by the number of code types, is a measure of the structural diversity. The number of code types for the 4c5v and 3c3v database classes is shown in Table I. By considering code types with more than one member, we can determine the accuracy of using a code type to predict the ^{31}P shift. This accuracy is shown in Table I by the column labeled "av intracode range". The intracode range is the range, in parts per million, spanned by the shifts produced by the members of a code type. Code types with single members have been excluded from the averaging of these intracode ranges. For code types with many members, wide intracode shift ranges often occur. This is more frequent in the 3c3v class, whose shifts display greater variation with respect to their structural environment. By analyzing the codes, we have obtained substantial evidence for β substituent effects occurring in both classes.

We have also carried out similar analyses but used the α substituent indexing code alone, and compared these results to those from the combined code. Including the β substituents does reduce the intracode ranges but, as would be expected, increases the number of code types. If, as in ^{13}C NMR, ^{31}P shifts are also influenced by γ and δ substituents, this would account for the wide intracode ranges seen in Table I.

Shift prediction methods based on substructure codes require each code type to have many members. This gives some statistical validity to the technique. It also allows the identification and elimination of errors. In the ACCESS system, some 15% of the HOSE code database was omitted by eliminating the outliers within each code type.[22] In ^{13}C, accurate predictions usually require the inclusion of γ substituents. Given the wider shift range of phosphorus, it is likely that substituents will have greater influence on the shift values than in carbon. Thus to obtain any substantial increase in predictive accuracy, we would need to include the effects of the γ and δ substituents. The resulting increase in the number of code types, however, would further reduce the statistical validity.

We have carried out other analyses on aspects of the structure ^{31}P shift relationship. One such analysis considers compound pairs. In these pairs the topology differs in only a single α substituent. The transition from one substituent to another is accompanied by a change in the shift. For any identical transition studied, the shift change is not constant across the range of compound pairs. There is, however, a strong correlation between the topology of the rest of the compound and the size and magnitude of a transition's shift change. This suggests that a substituent's effect on the shift is, in turn, affected by the topology of the rest of the structure. This finding has an important implication for the type of neural network architectures we can use to learn the structure–shift relationship in that the interrelationships between the structural variables preclude us from using techniques such as shared weighting schemes[34] to reduce the network dimensionality.

Limitations on the Use of Substructure Codes: Lexicographic versus Structural Similarities. Prediction methods using substructure codes work well when the majority of code comparisons are on lexicographic equivalence. Here the equivalence of two codes is a good measure of their structural equivalence in the focal region. Normally these methods use large ^{13}C substructure code reference databases. For any unknown compound, there is a high probability that most of its substructure codes will occur in the reference database. Thus the majority of comparisons will be on equivalence. Substructure code methods work less well when the lexicographic similarity of two codes is used as a measure of their structural similarity. This is due to there being no direct correspondence between the lexicographic similarity of two element symbols, and the chemical similarity of the elements they represent. In addition, each symbol should ideally have a weighting factor accounting for the effects of its distance from the focus. At the *subsymbol* level, however, the actual elements the symbols represent have many physical properties which can be used to make similarity comparisons. As an example (and while not the only factor) the correlation between the electronegativity value of substituents and the focal shift is well-known.

In our data set, including γ and δ substituents in any substructure code would lead to a high number of code types with single members. If the substructure codes were used to predict shifts, the majority of code comparisons would be on similarity and not on equivalence. Thus, while our method does use a substructure code, this code is designed for its

symbols to be replaced by some physical property of the elements. This allows more valid comparisons on similarity between code types.

According to our model, additivity rule methods also focus at the subsymbol level. In these methods substituent symbols are replaced by single numeric values. These values are valid only within restricted structural domains. A single substituent value probably embodies the combined effects of several physical attributes of the substituent, with each attribute influencing the shift. An attribute is itself likely to be influenced by the rest of the structure. This would account for a single value for a substituent only being applicable in similar structural types.

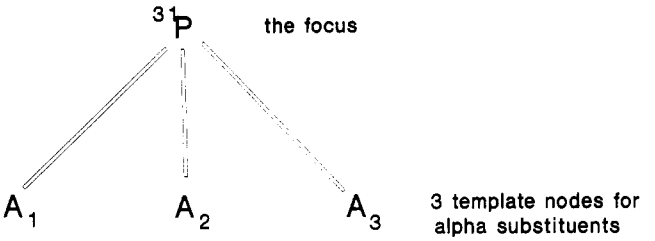
One strength of neural networks is their ability to extract and use any similarities in the input data that affect the output. The importance of similarities between the physical properties of elements, such as electronegativity, is well established. The ability of neural networks to learn continuous mappings[57,58] is also well established. This suggested they could be applied to learn the structure–shift relationship. Additionally, the inherent insensitivity of networks to individual data errors lends well to their use with large data sets. Our method inputs the structural parameters of a compound to a network, which then outputs a shift value. A network is trained using a set of compound vectors, each of which is composed of an input and an output vector. The input vector consists of a substructure code of a compound, with its element symbols replaced by some physical value. The output vector is a single value corresponding to the compound's ^{31}P shift.

Our original intentions were to replace the element symbols with their row and column values from the periodic table. We have initially used a single replacement value, however, for the following reason. Many problems solved by neural networks have been shown to be highly dependent on network architecture. Thus we anticipated exploring a range of such architectures. The effective size of our data set is small and limits the dimensionality of architectures we can *confidently* use. Replacing each symbol with two values doubles the size of the input vector. This further restricts the number of architectures we can use. As the correlation between electronegativity and shift values is well established, we can demonstrate the viability of the method by using electronegativity as symbol replacements.

EXPERIMENTAL SECTION

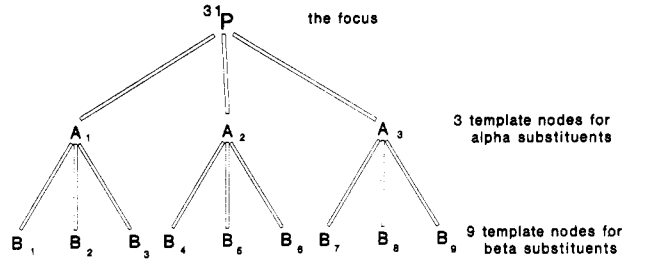
Nonspecialized neural networks require a set of vectors to contain a fixed number of components. As the structure of compounds varies considerably, this means some method of standardizing the size of compounds is required. Molecules are often represented as labeled graphs, with the graph nodes corresponding to atoms and the arcs to bonds. To standardize the size of compounds, we define a larger *template* graph, which contains a fixed number of nodes. A new graphical representation for a compound is then obtained by translating its molecular graph onto this template. As a template representation is abstracted from the molecular graph, we call such templates molecular abstract graph-spaces or MAGS.

Molecular Abstract Graph-Space Template(s). Different template formats are used with different phosphorus coordination classes. When a molecular graph is translated onto a template, the atoms of the molecule *occupy* the template nodes. As molecules contain different numbers of atoms, in some template representations many of these template nodes will be unoccupied. The template representation of a compound may, therefore, be sparsely populated.



A_{1-3} template nodes that can contain alpha substituents

Figure 2.



A_{1-3} template nodes that can contain alpha substituents

B_{1-9} template nodes that can contain beta substituents

Figure 3.

The format of a template graph is defined by three template variables. The first variable has a value equal to the degree or coordination number of the ^{31}P focus in the database class. This defines the number of template nodes that will be occupied by α substituents. The templates used with the 3c3v and 4c5v classes have first variable values equal to 3 and 4, respectively. The second template variable defines a *fixed* degree for the template nodes. This value, minus 1, defines the maximum allowed number of substituents for every *nonfocal* atom in the molecule. During translation, an atom's substituents are processed together and translated onto a template unit. The number of template nodes in a unit is equal to the second variable value minus 1. If the number of substituents is greater than this value, the translation process is terminated for that compound. For our two data sets the optimal value for the second template variable is 4. This excludes less than 20 compounds from translation in each class. The third template variable is the distance from the focus to which *focal* substituents are considered for translation. If a template has a third variable of value 1 then only α substituents are translated. We have mainly used templates onto which α, β, and γ substituents are translated. The value of the third variable in these templates is 3.

Figures 2–4 show three templates where only the value of the third template variable is altered. These templates all have a first variable value of 3 and a second variable value of 4. The ^{31}P focus is not translated onto a template. Figures 2–4 however *show* the focus for clarity. The Figure 2 template has a third variable value of 1, and thus only the α substituents will be translated. This template has a total of three nodes. The template in Figure 3 has a third variable value of 2, and α and β substituents will be translated. Compared to the Figure 2 template, this template has nine additional nodes, which can be occupied by β substituents, if any. The nine additional nodes correspond to the addition of three template units. As the second template variable has value 4, each unit consists of three template nodes. The template in Figure 4 can accommodate α, β, and γ substituents. A template unit is added to each of the nine nodes that may be occupied by

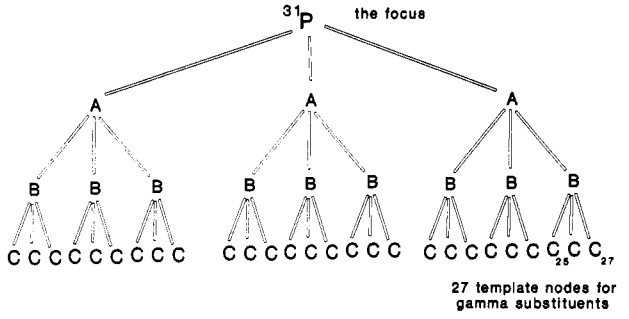


Figure 4.

a β substituent. Thus this template has 27 nodes which may be occupied by γ substituents. The total number of nodes in this template is 39.

The template method of standardizing compound size also allows standardization between coordination and valency classes. Compounds in the 3c3v and 4c5v classes can be compared by translating 3c3v compounds onto a template with a *first* variable value of 4. In such a template, each 3c3v compound will have one of the nodes allocated to the α substituents *unoccupied*. A network could then be trained using this mixture of the classes. We have not (as yet) investigated training networks with such mixtures.

During translation, an ordering is applied to the substituents of every atom. This ordering is based on the topological properties of a compound's molecular graph. This amounts to ordering a compound on its graph properties in the focal region. This topological information is incorporated into the template representation, and into the input vector, given to the neural network.

Inclusion of Problem Knowledge in the Input Vector. The ability of neural networks to learn classification or mapping tasks depends on a number of factors, such as the network architecture and the amount of problem knowledge given to the network. The knowledge given to the network is contained in the input vector. For some problems a network will succeed at its task given a minimum of knowledge. In other problems the omission of certain knowledge will cause it to fail at its task. Exactly which knowledge is crucial to solving a problem using a neural network is usually identified by expert(s) in the problem area. We wish the network to learn to predict NMR shifts from molecular topology. We therefore need to give the network as much information as possible about the topology. One aspect of this topological knowledge concerns the location of hydrogen atoms in the molecules. Another concerns the molecular topology in the focal region.

To reduce storage space, the adjacency matrices of chemical databases do not normally explicitly contain the positions of hydrogen. These are instead implied, by filling in the residual valencies of matrix atoms with hydrogen. This implicit definition of hydrogen is also used when many algorithms are applied to molecular graphs. The template representation of most compounds is sparsely populated. If hydrogen is not included in this representation, then no distinction can be made between unoccupied nodes and those that should be occupied by hydrogen. To ensure that the hydrogens are translated onto the template, we use an adjacency matrix where the hydrogen positions are explicitly defined.

The second aspect of topological knowledge concerns the *ordering* of a compound's atoms on the properties of the compound's molecular graph. Many such graph ordering algorithms exist.[68-77] These are often based on theoretical considerations and not on experiment significance. We have used the basic Morgan algorithm[78] to order the atoms in compounds. This ordering is applied to hydrogen explicit adjacency matrices. The significance of this ordering process is in the comparisons it allows *between* template represented compounds. Across a set of such representations, atoms occupying equivalent template nodes are equivalent with respect to the topology of the molecule in the focal region. Thus, intermolecular comparisons can be made between the atoms occupying equivalent nodes. One obvious piece of topological knowledge that we have omitted is the order of the bonds in compounds. We plan to investigate the effects of including this later.

After translation, the template representation is in turn translated into a substructure code. This code preserves the relative positions of atoms in the template representation. When referring to the substructure code, we will talk of the *positions* that atoms occupy in such a code. Since a substructure code is derived from a template representation, we term it a MAGS substructure code. Unless stated otherwise, the term *code* hereafter now refers to these MAGS substructure codes.

Template Translation and Generation of Substructure Code(s). The starting point for substructure code generation is the Molecular Design Ltd SDF file. This file contains a compound's adjacency matrix and other data such as the compound's ^{31}P shift value. The SDF file is output from the ChemBase database. To facilitate the checking of valency errors, each compound in our database has all its hydrogen atoms explicitly defined. One exception is for those hydrogens attached to carbon atoms, where the hydrogens are implicitly defined by the residual valencies of the carbons. The positions of these implied hydrogens are calculated, and a fully hydrogen explicit matrix is then produced. The Morgan algorithm is applied to this matrix, generating an extended connectivity (EC) score for every atom in a compound. These EC values are used to order substituents during their translation onto a template.

Translation starts from, but does not include, the ^{31}P focus. First the α substituents are ordered and translated onto the template. Next the β substituents are translated, starting with the β substituents attached to the highest scoring α substituent. The templates we have used for each class of compound have all the template nodes allocated to α substituents occupied. Apart from the focus, an atom may have less actual substituents than the number of template nodes allocated to its substituents. If this is the case, then some of these allocated template nodes are deemed *unoccupied*. Substituents are translated onto a unit, and any node in the unit that is deemed unoccupied cannot be filled later. For the templates shown in Figures 2–4, translation will process a unit of three template nodes at a time. For the Figure 2–4 templates, translation terminates when the last template unit has been processed. For the Figure 4 template, termination would occur once nodes C_{25} to C_{27} had been allocated a substituent or deemed unoccupied.

The upper section of Figure 5 illustrates the procedure for any atom Z in a compound. Atoms A–C are substituents of Z. The numbers underneath A–C are their EC values. Atom X is nearer to the ^{31}P focus than Z, and thus Z is considered to be a substituent of X. X and Z have already been translated onto the template. Translation is at the point just before the substituents of atom Z are translated. The template nodes

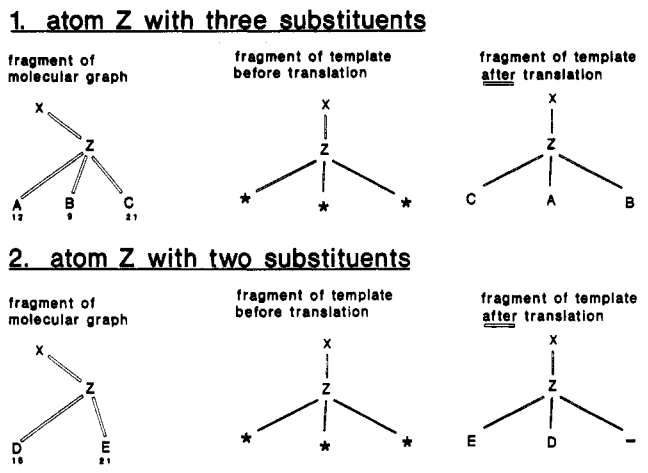


Figure 5.

in the unit allocated to the substituents of Z have not yet been processed, and are marked with an asterisk. Translation orders A–C on their EC scores. They are then translated and occupy the allocated template nodes. The procedure for atom Z with only two substituents D and E is also illustrated in the lower part of Figure 5. Here the symbol "-" indicates the template node is unoccupied. If two substituents of an atom have identical EC scores, then this is resolved by considering additional information on the substituents, such as their valency and charge.

After translation of a compound, the template representation is then converted into a substructure code. In addition to the substructure code, three other items of data on the compound are output. These additional compound data are contained in the SDF file. These are (i) the database identity number of the compound, (ii) an ordered list (smallest first) of the sizes of rings containing the ^{31}P focus, and (iii) the ^{31}P NMR shift.

Figure 6 shows a compound with its α, β, and γ substituents numbered, its template representation, and the substructure code derived from the template. The Figure 4 template containing 39 nodes has been used. The substructure code therefore has 39 *positions*. The partition of the substructure code shown in Figure 6 into α, β, and γ substituents is shown below. In the substructure codes, the symbol "-" signifies that the corresponding template node is unoccupied.

3 α substituents	9 β substituents	27 γ substituents
CCC	PSi-HHHHHH	CCCCCC- - - - - - - - - - - - -

In this substructure code, positions 1–3 are occupied by α substituents, and positions 4–12 by β substituents, if any. Positions 13–29 are occupied by γ substituents, if any. Below is shown the location of the numbered atoms of the structure in Figure 6 in the substructure code.

α substituents	$C_1\ C_2\ C_3$
β substituents	$P_4\ Si_5$ - $H_6\ H_7\ H_8\ H_9\ H_{10}\ H_{11}$
γ substituents	$C_{12}\ C_{13}\ C_{14}\ C_{15}\ C_{16}\ C_{17}$ - - - - - - - - - - - - - - -

By lexicographically sorting such substructure codes, we can obtain supporting evidence for (a) the influence of γ and δ substituents on the ^{31}P shift and (b) the moderation of substituent effects by the complete molecular topology. Table II shows some of this evidence from the 3c3v class. The substructure codes in Table II have been generated from the template shown in Figure 4. The occurrence column is the number of occurrences of the substructure code type in the

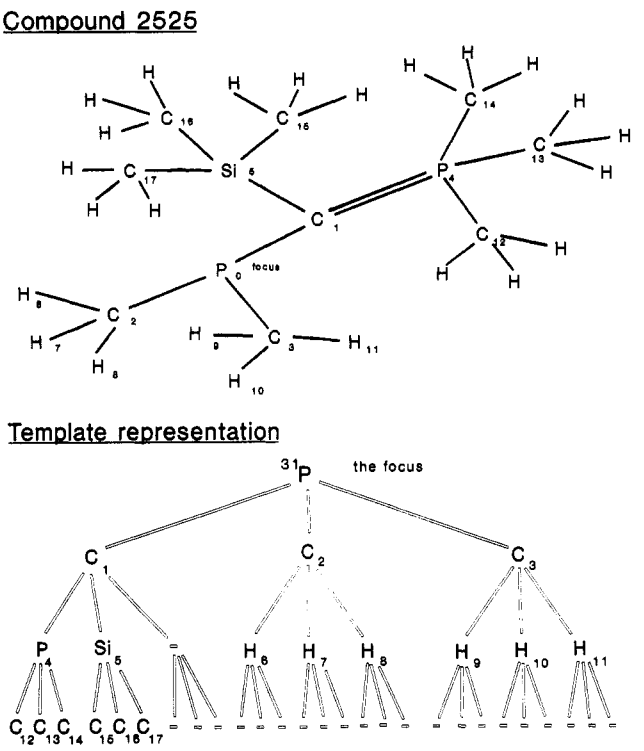


Figure 6.

database. The upper and lower shift columns are the limits of the shifts of the members of that code type. Within each section in Table II, structures have similar topology in the focal region. In section 1, replacing fluorine with hydrogen results in a positive displacement of shift. In section 2, the same replacement results in the shift being negatively displaced. In section 3 the shift shows no displacement.

Dealing with Cyclic Structures. Our analysis of cyclic compounds is restricted to ring systems containing the ^{31}P focus. Comparisons are made to acyclic structures which are the most topologically similar in the focal region. These comparisons have been carried out using the Handbook's indexing codes and the MAGS substructure code(s). A summary of these results follows. The effects of rings on the shift appear to be negligible for ring systems with more than seven members. For five- and six-membered rings the shift is displaced positively and negatively, respectively, with an average magnitude of 10 ppm in each case. The few three-membered ring systems in the database always exhibit large negative displacements of the shift. There are few four-membered ring systems, and the evidence from these is contradictory. These shift displacements by ring systems may arise purely from their effects on the bond angles at phosphorus. The fact that two bond paths occur from ring substituents to the focus may, however, also be relevant. Currently we can only comment that the majority of these ring effects appear to be consistent.

To give the neural network information on ring systems containing the ^{31}P focus, we use a five-component ring vector. This is added to the substructure code to form the input vector. In the input vector, the ring vector precedes the substructure code. The format of these ring vectors is shown in Table III. As smaller rings appear to have greater influence on shifts, the ring vector for three-membered rings is five 1s. This decreases to four zeros and a single 1 for seven-membered

Table II

ring data	shift av	focal substituents (EC ordered) α	β	γ	occurrence (no. of members of code type)	highest shift of a member	lowest shift of a member	range of shifts of members
				Section 1				
0	113.5	CBrBr	CC-------	CF-CF----------	1	113.50	113.50	0.00
0	135.9	CBrBr	CC-------	CF-CH----------	1	135.90	135.90	0.00
0	149.5	CBrBr	CC-------	CH-CH----------	10	157.00	141.70	15.30
0	13.0	CCBr	CC-CC----	CF-CF----CF-CF----------	1	13.00	13.00	0.00
0	39.3	CCBr	CC-CC----	CF-CF----CH-CH----------	1	39.30	39.30	0.00
0	72.9	CCBr	CC-CC----	CH-CH----CH-CH----------	1	72.90	72.90	0.00
0	-78.5	CCC	CC-CC-CC-	CF-CF----CF-CF----CF-CF---- (6 F)	2	-78.50	-78.50	0.00
0	-48.7	CCC	CC-CC-CC-	CF-CF----CF-CF----CH-CH---- (4 F)	1	-48.70	-48.70	0.00
0	-44.6	CCC	CC-CC-CC-	CF-CH----CF-CH----CF-CH---- (3 F)	1	-44.60	-44.60	0.00
0	-32.5	CCC	CC-CC-CC-	CF-CH----CF-CH----CH-CH---- (2 F)	1	-32.50	-32.50	0.00
0	-26.3	CCC	CC-CC-CC-	CF-CF----CH-CH----CH-CH---- (2 F)	1	-26.30	-26.30	0.00
0	-19.8	CCC	CC-CC-CC-	CF-CH----CH-CH----CH-CH---- (1 F)	1	-19.80	-19.80	0.00
0	-6.8	CCC	CC-CC-CC-	CH-CH----CH-CH----CH-CH---- (0 F)	54	6.80	-20.30	27.10
0	37.0	CCCl	CC-CC----	CF-CF----CF-CF----------	1	37.00	37.00	0.00
0	57.1	CCCl	CC-CC----	CF-CF----CH-CH----------	1	57.10	57.10	0.00
0	100.7	CCCl	CC-CC----	CH-CH----CH-CH----------	4	167.40	74.70	92.70
0	-143.0	CCH	CC-CC----	CF-CF----CF-CF----------	1	-143.00	-143.00	0.00
0	-92.2	CCH	CC-CC----	CF-CF----CH-CH----------	1	-92.20	-92.20	0.00
0	-42.6	CCH	CC-CC----	CH-CH----CH-CH----------	11	-40.20	-45.40	5.20
0	137.0	CClCl	CC-------	CF-CF----------	1	137.00	137.00	0.00
0	147.9	CClCl	CC-------	CF-CH----------	1	147.90	147.90	0.00
0	160.0	CClCl	CC-------	CH-CH----------	10	166.00	152.00	14.00
				Section 2				
0	12.8	SCC	C--FFFFFF	FFF----------	1	12.80	12.80	0.00
0	37.1	SCC	C--FFFFFF	HHH----------	1	37.10	37.10	0.00
0	14.1	SeCC	C--FFFFFF	FFF----------	1	14.10	14.10	0.00
0	27.9	SeCC	C--FFFFFF	HHH----------	1	27.90	27.90	0.00
				Section 3				
0	27.0	SCC	C--FFFHHH	FFF----------	1	27.00	27.00	0.00
0	27.0	SCC	C--FFFHHH	HHH----------	1	27.00	27.00	0.00

Table III. Ring Vector Encodings of (^{31}P Containing) Ring Sizes

ring size	ring code	ring size	ring code
3	1 1 1 1 1	6	0 0 0 1 1
4	0 1 1 1 1	7	0 0 0 0 1
5	0 0 1 1 1	0	0 0 0 0 0

rings. Ring sizes greater than seven are treated as acyclic and have a ring vector of five zeros. Currently our input vector only includes one ring vector per compound. If a compound has more than one ring system containing ^{31}P, then the ring vector for the smallest ring is used. An input vector derived from the template in Figure 4 will have a total of 44 components, 5 from the ring vector and 39 from the template.

During the translation of a molecular graph onto a template, a history list of the atoms already translated is kept. *After* the substituents of the current atom are ordered, they are matched against the contents of this list. If a substituent of the current atom is already in the list, this indicates a cyclic pathway. The second or "current" occurrence of the substituent atom is not transferred to the template, and the node it would have occupied is deemed unoccupied. Thus each atom in a compound only occurs once in the template.

Generation of Input Vector Sets. Prior to generating input vectors from the ring information and substructure code, a further processing stage is required. This involves the elimination of any *one to many* mappings from the set of compounds. The source of these mappings is code types with more than one member. The third substructure code in Table II, section 1, with 10 members, is an example of this. Our initial experiments have shown that if these duplicates are not eliminated, the network will oscillate. The elimination of these one to many mappings creates a data set where each code type only occurs once. For any code types in the original data set which have more than one member, an average shift value is calculated. The set of unique code types can now be used to generate the set of compound vectors for the neural network. A compound vector is composed of three parts: (i) the ring vector from Table III, (ii) the MAGS substructure code with the symbols replaced by physical values (Allred–Rochow *scaled* electronegativity values in this paper), and (iii) the *scaled* ^{31}P chemical shift of the compound. Parts i and ii form the input vector part of the compound vector, and part iii forms the output vector.

Our neural network architectures use the sigmoidal transfer function ($1/[1 + e^{-NET}]$) throughout. The shift range that a data set spans must therefore be scaled to lie within the 0–1 output range of the transfer function. The 3c3v class spans a range of some 750 ppm (–455 to +295 ppm). This shift range is scaled to 0.1–0.9, and the shifts of compounds are scaled accordingly. The range of 0.1–0.9 is used as it lies in the most linear region of the output range of the transfer function. We also scale the (Allred–Rochow) electronegativity values used as symbol replacements. These are scaled from the range of 0–4.2 to lie in a range of 0–0.9. We scale these values to have a common input range for all of the different physical properties of an element that we wish to compare with respect to prediction performance.

Figure 7 shows the data that are output from the processing of compound 2525 shown in Figure 6. For brevity, we have only shown the α and β substituents in Figure 7. This corresponds to the template in Figure 3 being used. Also shown in Figure 7 are the components of the compound vector, where (a) the ring data are replaced by their ring vector, (b) the symbols of the substructure code are replaced by their (Allred–Rochow) electronegativity values, and (c) the compound's ^{31}P shift value of –44.2 is *scaled* to lie on the range of 0.1–0.9. For reasons of brevity we have shown the *full* electronegativity values in Figure 7 and not the *scaled* values

All the data output on compound 2525

identity number	chemical shift	ring data	substructure code
2525	-44.2	0	C C C P Si - H H H H H H

Components of the compound vector from compound 2525

1. the ring vector: 0 0 0 0 0

2. the substructure code replaced by electronegativity values: 2.5 2.5 2.5 2.06 1.9 0 2.2 2.2 2.2 2.2 2.2 2.2

3. the scaled chemical shift value: 0.5391884

Figure 7.

actually used.

Why Two Monitoring Sets Are Generated. Before training a neural network, we divide the set of compound vectors into three. One set, called the *training* set, is used to train the network. The other two *monitoring* sets are used to evaluate the trained network. Trained networks are evaluated by measuring their performance in predicting the ^{31}P shifts of compounds in the monitoring sets. We generate two monitoring sets in *anticipation* of a phenomenon known as overtraining. This phenomenon *may* occur during training. A network overtrains when it starts considering characteristics peculiar to only the training set. These characteristics are not relevant to the network's prediction performance on the other data set(s). When a network overtrains, its prediction performance on the training set vectors increases, but its performance on the monitoring set(s) vectors decreases. To find the point at which a network starts to overtrain, the network is usually monitored during training on an *evaluation* set. Training is stopped when the prediction performance on this set starts to decrease. As the evaluation set data are used during training, a second set of completely unseen data is required for testing. This is the second monitoring or *test* set.

Currently we do *not* use an evaluation set to interactively monitor a network during training. Instead a network is trained for a specified number of epochs and the value of the weight vector is saved every epoch. (An epoch is one cycle through all the compounds in the training set.) After training is finished, the prediction performance at each epoch is then calculated using each monitoring set *separately*. This allows the full automation of the training and monitoring process. We adopted this approach after our initial trials, which trained networks to 40 000 epochs. These trials showed that most learning took place within the first 1000 epochs. From 1000 to 40 000 epochs we can detect *no* substantial increase or decrease in the prediction performance. Thus our networks do not appear to exhibit the overtraining phenomenon. This is probably due to the current crudeness of our method. The sources of this crudeness are discussed later. We continue to use two monitoring sets as this allows overtraining to be detected. We anticipate that our networks will exhibit overtraining as our method is refined.

The measure of prediction performance that we use is the percentage of compounds that have a network predicted shift within a given tolerance of the actual shift. These tolerances are measured in *actual* parts per million. The measure of network error that we use is an average obtained from all the compounds in the set being tested. An individual error from a compound vector is calculated by taking the absolute value of [network predicted shift − output vector shift]. These values are the *scaled* values of the shift(s) and lie in the 0.1–0.9 output range of the transfer function.

Table IV. Characteristics for the MAGS Substructure Code Generated from the Template in Figure 4[a]

class	no. of compds	no. of code types	range of class, ppm	network set sizes: train	evaluate	test
3c3v	3512	2248	749.2	1748	250	250

[a] These codes include α, β, and γ substituents.

Table V. Characteristics for the MAGS Substructure Code Generated from the Template in Figure 3[a]

class	no. of compds	no. of code types	range of class, ppm	network set sizes: train	evaluate	test
3c3v	3512	1260	749.2	960	150	150

[a] These codes include α and β substituents.

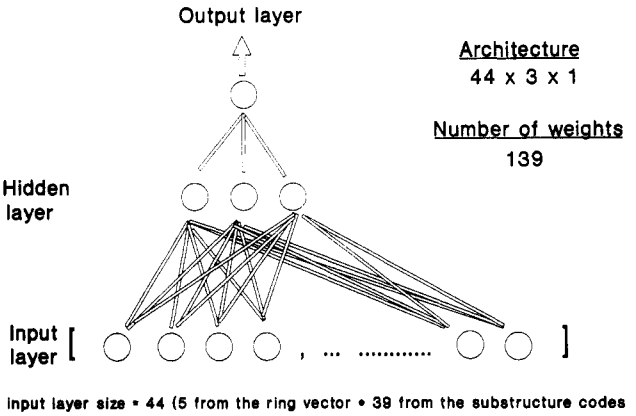


Figure 8. Fully connected 44 × 3 × 1 network.

Tables IV and V show the number of compound vectors we have allocated to each of the three sets. In each table, column 3 shows the number of substructure code types in the 3c3v class. The range that the shifts of the 3c3v class spans in parts per million is given in column 4. The sizes of the training and monitoring sets are shown in the "network set sizes" column(s). The Table IV data come from substructure codes generated using the Figure 4 template, and the data in Table V, from codes generated using the Figure 3 template. Table IV has 74 compounds less than Table I. This is due to our inability to translate unambiguously the Handbook representation of these 74 compounds. The Handbook represents molecular topology by a line formula.

Network Construction and Architectures. As previously mentioned, the neural network architecture used is crucial in many problems. Thus we anticipated exploring a range of network architectures. We have chosen to use the MITRE Corp. simulator package[79] to generate network architectures. This system allows architectures to be quickly generated and modified. An architecture is specified by a high level declarative language known as ASPIRIN. Figure 8 illustrates the architecture of one of the networks used with the 3c3v set. This network has 44 input neurons, 3 hidden layer neurons, and 1 output neuron. The network contains 139 weights ((44 × 3) + (3 × 1) + 4 bias weights). This network is therefore used with compound vectors which are derived from a Figure 4 template and contain only one ring vector.

We have evaluated architectures containing 0–2 hidden layers and also architectures with different hidden layer sizes. All the networks we have used are fully connected, with the noninput-layer neurons having an input bias. The learning rate and momentum coefficients used are kept constant at

0.02 and 0.5, respectively. These learning rate and momentum coefficients can be highly specific to a problem. There is, however, a general consensus on the range of values one would normally initially apply. The learning rate of 0.02 is somewhat lower than might be expected. To explain why we have initially used such a low learning rate, we first elaborate on what is occurring during the training process.

The network can be viewed as consisting of an N-dimensional weight space. Another dimension, that of the network error, can be added. Every value of the vector of network weights has an associated error value. This allows the conceptualization of an error *surface* above the network's weight space.[80] During training, the network weights are altered to reduce the overall error on the training set. This *gradient descent* process ideally takes the weight vector on a path downward in the error surface toward a point with a minimum value for the error. Perceptrons, i.e. networks with no hidden layers, have bowl-shaped error surfaces and only one minimum point in their error surface (if such a minimum occurs). Networks with hidden layers are essentially formed by concatenating one or more Perceptron architectures. The resulting concatenation of the error surfaces can produce a highly convoluted error surface. This convoluted surface may contain *local* minima. In these multilayer networks if the weight vector oscillates on the error surface, it can get stuck in a local minimum. This phenomenon is known as local minimum entrapment. The momentum term is used during training to help reduce any oscillations of the weight vector.[81] Substantial oscillations may, however, still occur if two compound vectors move the weight vector in *opposite* directions on the error surface. There is a very high probability of this occurring within our data set. This is due to some compounds having the wrong sign for their shifts. The reasons for this have been mentioned previously.[67] Vectors derived from compounds with sign errors will move the weight vector in a different direction from the general trend. Thus to dampen these *expected* oscillations, we have used a low learning rate.

The reason why the network oscillates when trained with a set containing one to many mappings should now be clear. These one to many mappings are subsets which have the *same* input vector, but a different output vector. These subsets therefore cause the weight vector to move in different directions on the error surface.

Network Controls and Input Vector Types. We train our networks using four different variations on the input vector part of a compound vector. In each of these input vector types, the ring vector component is unaltered. The variations are therefore restricted to the part of the input vector containing the substructure code. In all input vector types, unoccupied positions in the substructure code are replaced with zero. Two of the input vector types are controls.

The first control has each element symbol in the substructure code replaced with a value of zero. Thus the entire substructure code part of the input vector will consist of zeros. We term this a *zero* input vector. This allows us to determine how well the network performs given only the information contained in the ring vector.

The second control has the substructure code symbols replaced by their scaled Allred–Rochow electronegativity values. The positions in the substructure code are then scrambled. This destroys the topological knowledge built into the substructure code part of the input vector. The scrambling of positions is carried out in the training set *only*. We term this control a *random* input vector. This allows us to determine what effect the ordering of compounds on their molecular

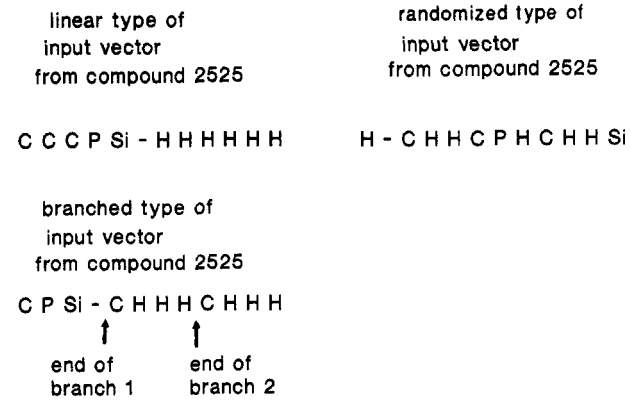


Figure 9. Input vector types (substructure code part, symbolic form).

graphs has on performance. Our scrambling procedure uses the randomization method described in Knuth.[82] One possible result of this randomizing of the substructure code positions is shown in Figure 9. For clarity we have shown the effects of scrambling on the symbolic form of the substructure code part of the input vector.

The two other types of input vector are structured input vectors. Here the adjective "structured" refers to the substructure code part of the input vector. Across a set of the same type of structured input vector, the positions in the substructure code part are equivalent with respect to the graph properties of the molecules. The substructure codes we have shown so far, in Table II and in Figures 6 and 7, are used to generate a *linear* input vector. This is done by straightforward replacement of the symbols by their scaled Allred–Rochow electronegativity values. The other type of structured input vector we call a *branched* input vector. The branched and linear input vectors differ in the substructure code positions onto which the template nodes are translated. The branched input vector is derived from a template by considering each node containing an α substituent to be the start of a branch. All the substituents in a branch are translated onto the (branched) substructure code first. The next branch is then considered. Figure 9 shows the symbolic form of the branched input vector derived from compound 2525.

RESULTS

The results from training networks with compounds in the 3c3v class are shown in Tables VI and VII. The results in Table VI were obtained using the four types of input vectors derived from the template of Figure 4. Thus α, β, and γ substituents are considered. Table VII shows the results from training networks using linear input vectors derived from the template of Figure 3. Thus only α and β substituents are considered. In both Tables VI and VII the percentage of correct predictions *within* the given tolerances after 4000 epochs is shown in columns 3 (within 20 ppm) and 4 (within 40 ppm). Columns 5 (greater than 80 ppm) and 6 (greater than 100 ppm) show the percentage of predictions where the difference is *greater* than the tolerance value. In calculating into which tolerance band a predicted shift falls, the absolute value of [network predicted shift – actual shift] is taken. We have found only minor differences between performance on the testing and evaluation sets. We have therefore included both monitoring sets in the averaging of the results. The results are an average for five *different* partitions of the data into training, test, and evaluation sets, with the following exceptions: (i) Table VI, linear input vector, 20 different partitions; (ii) Table VII, linear input vector, 10 different partitions.

Table VI. Results from the 3c3v Set with Input Vectors Generated Using the Template of Figure 4

architecture of neural network used	input vector type used	performance after 4000 epochs (performance after 1 epoch)			
		% below value of tolerance		*% above* value of tolerance	
		20 ppm	40 ppm	80 ppm	100 ppm
44 6 2 1	linear	48 (10)	73 (21)	8 (50)	4 (36)
44 6 2 1	branch	49 (10)	73 (21)	8 (52)	4 (38)
44 6 2 1	zero	11 (9)	24 (19)	42 (50)	31 (34)
44 6 2 1	random	15 (9)	28 (20)	48 (52)	37 (37)
44 10 1	linear	48 (10)	73 (21)	8 (49)	4 (35)
44 10 1	branch	50 (10)	74 (21)	8 (52)	4 (38)
44 10 1	zero	11 (9)	24 (19)	42 (51)	31 (34)
44 10 1	random	15 (9)	29 (19)	47 (52)	38 (36)
44 3 1	linear	48 (10)	73 (21)	8 (50)	4 (35)
44 3 1	branch	47 (10)	73 (21)	8 (52)	5 (37)
44 3 1	zero	11 (9)	24 (19)	43 (51)	31 (34)
44 3 1	random	14 (9)	28 (19)	47 (52)	35 (37)
44 1	linear	37 (21)	62 (41)	13 (29)	8 (20)
44 1	branch	35 (20)	60 (39)	13 (28)	8 (23)
44 1	zero	11 (10)	24 (22)	43 (46)	31 (34)
44 1	random	13 (14)	27 (27)	45 (50)	33 (38)

Table VII. Results from the 3c3v Set with Input Vectors Generated Using the Template of Figure 3

architecture of neural network used	input vector type used	performance after 4000 epochs (performance after 1 epoch)			
		% below value of tolerance		*% above* value of tolerance	
		20 ppm	40 ppm	80 ppm	100 ppm
17 3 2 1	linear	33 (10)	61 (24)	12 (51)	6 (40)
17 3 1	linear	31 (10)	60 (24)	12 (51)	7 (40)
17 1	linear	29 (13)	55 (29)	18 (43)	11 (31)

Tables VIII and IX show the results obtained by averaging the evaluation and test sets separately. These tables also show the maximum and minimum performance values obtained in these sets at 4000 epochs. The results in Tables VI–IX have been rounded up or down to the nearest whole number.

The rapid rate of increase in performance using the structured codes is better demonstrated graphically. Therefore we have included Figures 10 and 11. The graphs in these figures are obtained from using networks with 44 × 3 × 1 architectures. The data shown in these graphs come from the same experiments as the data in Table VI. Figure 10 shows the results of training with linear input vectors derived from the Figure 4 template. Figure 11 shows the results of training with random input vectors derived from the Figure 4 template. The right y axis of the graphs in Figures 10 and 11 shows the decrease in the network error as training progresses. The left y axis shows the percent of compounds with predicted shifts in the given tolerance regions of Table VI. The x axis shows the epoch number.

If networks are trained using linear input vectors and are evaluated with a set of branched input vectors (or the converse), then the performance levels are at or below those of the controls. These networks still exhibit increases in performance comparable to Figure 10 if evaluated with the same type of structured vector they were trained with. Similar differences in performance between structured input vectors and the controls are obtained in the 4c5v set.

DISCUSSION

Networks trained with either type of structured input vector show considerable increases in performance compared to the controls. This is more obvious if this performance is shown graphically, as in Figure 10. Here the rapid increase of the number of compounds in the below 20 and 40 ppm (tolerance) bands is accompanied by decreases in the *over* 80 and 100 ppm bands. Concomitantly the average network error decreases. The importance of including topological knowledge in the input vector is shown by comparing the performance of the two controls. Networks trained with zero vectors have only the knowledge contained in the ring vector. Random vectors have the ring vector knowledge *and* that of the scaled electronegativity values which replace the substructure code symbols. Across the set of random vectors, however, the input vector positions are *not* equivalent with respect to the molecular topology. Figure 11, the results of training with random input vectors, shows little increase in performance.

The importance of the equivalence of positions in the input vectors of a data set is further underlined by cross evaluation experiments. Across each set of the two different types of structured vector the input vector positions are equivalent with respect to the molecular topology. Between the sets, apart from the first position (highest EC scoring α substituent), they are *not* equivalent.

We conclude that training with structured input vectors allows the network to learn to correlate the electronegativity values of substituents and the value of the shift. We also conclude that the information relating to the topology of compounds in the focal region is crucial to the ability of a network to learn or find this correlation.

Table VII shows the results of training a network using linear input vectors containing α and β substituents alone. The performance is markedly reduced compared to results from using the equivalent input vector type shown in Table VI. As Table VI vectors include γ substituents, this suggests that the inclusion of γ substituents increases the prediction performance.

Architectures with hidden layers *always* outperform those with none. As Perceptrons can only learn to simulate linear functions, this suggests that the structure–^{31}P shift relationship has both linear and nonlinear components. Coincidentally, in ^{13}C NMR, nonlinear correction factors have recently been incorporated into linear additivity rule systems.[15] These corrections cater to nonlinear relationships observed in compounds with more than one highly electronegative α substituent. Figure 10 shows that the most rapid increase in performance occurs in the 0–400 epoch region. All architectures used in both classes show this rapid early increase, allowing for the differences imposed by the *dynamics* of the architecture. The Perceptron architectures show higher values at 1 epoch than their multilayer counterparts. This is almost certainly due to learning taking place during the first epoch.

Table VIII. Results from the 3c3v Set with Input Vectors Generated Using the Template of Figure 4. Results from *Evaluation Set* Only

		performance after 4000 epochs											
		% *below* value of tolerance						% *above* value of tolerance					
		20 ppm			40 ppm			80 ppm			100 ppm		
architecture of neural network used	input vector type used	low	av	high	low	av	high	low	av	high	low	av	high
44 6 2 1	linear	42	47	54	66	74	78	5	7	10	2	4	7
44 6 2 1	branch	44	49	52	68	74	76	6	7	10	3	3	4
44 6 2 1	zero	9	11	14	21	24	28	40	42	44	29	31	34
44 6 2 1	random	11	14	16	23	27	29	40	49	58	30	37	45
44 10 1	linear	42	48	54	66	73	78	5	7	11	2	4	7
44 10 1	branch	44	50	53	70	74	80	6	7	8	2	3	4
4 10 1	zero	9	11	14	21	24	28	39	42	44	29	31	34
44 10 1	random	11	13	16	25	29	31	42	48	54	32	38	44
44 10 1	linear	42	47	54	67	73	78	6	8	12	2	4	7
44 3 1	branch	44	47	51	70	74	75	6	8	9	4	4	5
44 3 1	zero	9	11	14	21	25	28	40	42	44	30	32	34
44 3 1	random	8	14	20	21	27	32	44	48	55	31	36	42
44 1	linear	32	37	41	57	62	67	10	13	17	6	8	12
44 1	branch	30	35	42	57	62	67	10	12	14	4	7	9
44 1	zero	9	11	14	21	285	28	40	43	45	29	31	34
44 1	random	10	12	15	22	25	30	41	46	51	28	33	38

Table IX. Results from the 3c3v Set with Input Vectors Generated Using the Template of Figure 4. Results from *Test* Set Only

		performance after 4000 epochs											
		% *below* value of tolerance						% *above* value of tolerance					
		20 ppm			40 ppm			80 ppm			100 ppm		
architecture of neural network used	input vector type used	low	av	high	low	av	high	low	av	high	low	av	high
44 6 2 1	linear	41	49	57	67	73	78	4	8	10	2	4	6
44 6 2 1	branch	48	50	55	70	72	76	6	8	10	3	4	5
44 6 2 1	zero	9	11	12	20	23	28	37	43	48	26	31	36
44 6 2 1	random	14	16	17	28	30	34	44	46	47	33	35	40
44 10 1	linear	41	48	54	68	72	77	5	8	10	2	4	7
44 10 1	branch	48	50	56	70	74	77	7	8	9	3	4	5
4 10 1	zero	10	11	12	20	23	27	38	43	46	26	31	36
44 10 1	random	14	17	20	25	30	33	36	46	57	29	38	46
44 3 1	linear	43	48	52	68	73	77	4	8	10	2	4	7
44 3 1	branch	42	46	50	67	71	76	7	9	10	4	5	6
44 3 1	zero	9	11	12	20	23	28	37	43	48	26	31	36
44 3 1	random	12	14	16	25	28	31	38	46	51	28	33	37
44 1	linear	33	38	43	57	62	66	11	13	16	6	8	11
44 1	branch	32	35	38	56	60	63	12	14	17	5	8	11
44 1	zero	9	11	12	20	24	28	37	42	46	26	31	36
44 1	random	12	13	13	25	28	31	38	44	50	27	32	38

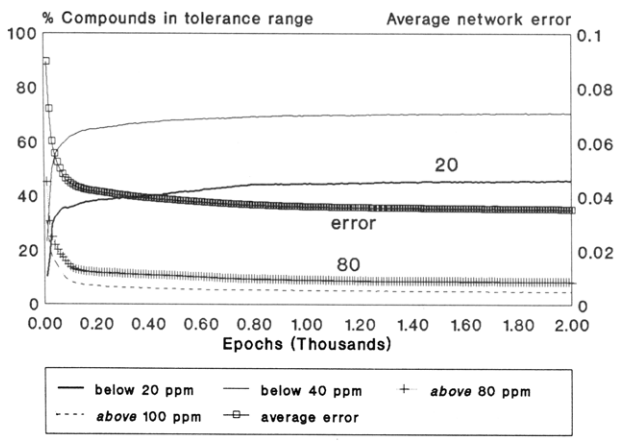


Figure 10. Linear input vector (3c3v class, architecture 44 × 3 × 1).

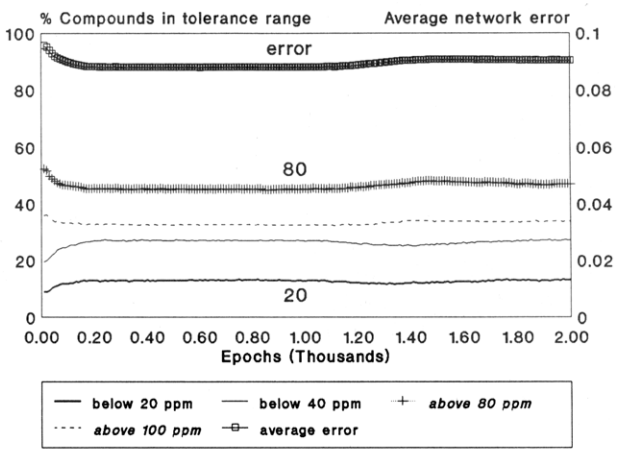


Figure 11. Random input vector (3c3v class, architecture 44 × 3 × 1).

The error surface for Perceptron architectures is smoother compared to the more convoluted error surfaces of the multilayer architectures. For mappings that the Perceptron *can* represent, i.e., linear mappings, the weight vector therefore has a shorter path to the minimum. thus we would expect Perceptrons to learn any linear component in a mapping faster than multilayer architectures, and this is what occurs. In the multilayer architectures, the slow rate of increase after the early region is probably due to the learning of the nonlinear component in the structure–shift relationship. There is little difference in overall performance between architectures using two hidden layers as opposed to one.

Currently our method is likely to be highly suboptimal. The sources of crudeness in the method arise from the factors outlined below.

(i) The error level in our data. These errors arise from many sources and are common in large databases. Due to the high structural diversity of the database, the identification of many of these errors is very costly. Errors in the training set, while not catastrophic, will lead to a reduction in the prediction performance. Errors in the monitoring sets will be perceived as a reduced performance.

(ii) The graph ordering procedure used. The Morgan algorithm ordering is known to be suboptimal on theoretical grounds. Between iterations, the EC values can oscillate substantially. Many more advanced ordering algorithms have been derived.[68-77] A comparison of the application of these algorithms, relative to the prediction performance is planned.

(iii) Use of sigmoid transfer functions. We have used the sigmoid transfer function supplied with the MITRE simulator. Studies comparing transfer functions indicate greater prediction performance can be obtained by using functions other than sigmoid.[83] Using networks with other types of transfer function is planned.

(iv) The ring vector used to give the network information on cycles containing the focus. This is only an educated guess at encoding such information and requires further study.

(v) Three-dimensional or "through-space" effects. The current method does not account for any three-dimensional or "through space" effects of substituents on the focal shift.

(vi) The omission of knowledge of the bond order of compounds. Networks trained with structured vectors reach a performance maximum. This may be due to our current omission of the bond order information from the input vector.

In neural network learning, difficulties are often encountered when a problem is scaled up from a prototype size to a practical one. This is one reason why we have initially used *all* the data in a class. Another is that it allows confident exploration of more network architectures. Using all the data with a single *static* replacement value, such as electronegativity, will not give the best predictions, for the following reasons. Throughout all the different structural types in a database class, we have represented substituents by their electronegativity value. In some types of structure, however, this value will be closer to the *optimum* value than in other types. Here, by an optimum value for a substituent, we mean one that would have been derived by an additivity rule approach. Thus we would expect these static electronegativity replacement values to give different performance results in different subclasses. Our method can be modified to allow the network *itself* to derive a set of optimum substituent values specific to these subclasses. This will allow the *automatic* derivation of parameters similar to those used in additivity rules from existing topological databases. These values will of course be stored in the network weights, allowing their automatic application.

As our method is based on the occurrence of α–δ substituent effects in ^{31}P NMR, its extension to other elements displaying these effects, such as carbon, should be relatively easy. Additionally, its extension to include molecular mechanics parameters should also be possible.

Obtaining networks limited to predicting shifts from specific substructural classes, however, is a short term goal. Given the ability of neural networks to extract the underlying similarities within a data set, and the physical similarities between elements, they should theoretically be capable of deriving a more general solution. This will however require the input of more physical parameters per symbol and will require a larger data set.

ACKNOWLEDGMENT

The author would like to thank the reviewers of this paper for their many helpful comments.

REFERENCES AND NOTES

(1) Buchanan, B. G.; Sutherland, G. L.; Fiegenbaum, E. U. In *Machine Intelligence 4*. Meltzer, B., Michie, D., Eds.; Edinburgh University Press: Edinburgh, 1969; pp 209–254.
(2) Bremser, W.; Fachinger, W. Multidimensional Spectroscopy. *Magn. Reson. Chem.* **1985**, *23*, 1056–1071.
(3) Robein, W. Computer-Assisted Structure Elucidation of Organic Compounds III*: Automatic Fragment Generation From ^{13}C-NMR Spectra. *Mikrochim. Acta [Wien]* **1986**, *II*, 271–279.
(4) Ditchfield, R.; Miller, D. P.; Pople, J. A. Self-consistent Molecular Orbital Methods XI: Molecular Orbital Theory of NMR Chemical Shifts. *J. Chem. Phys.* **1971**, *54*, 4186–4193.
(5) CNDO: Sadlej, A. J. Modified CNDO Methods II: Electronic Structure of Acetonitrile and Its Complexes with Metal Cations. *Org. Magn. Reson.* **1970**, *2*, 63–69.
(6) INDO: Herring, F. G. Chemical Shift I: Approximate Theory and Application to First Row Binary Fluorides. *Can. J. Chem.* **1970**, *48*, 3498–3503.
(7) Grant, D. M.; Paul, E. G. Carbon-13 Magnetic Resonance II. Chemical Shift Data for the Alkanes. *J. Am. Chem. Soc.* **1964**, *86*, 2984–2990.
(8) Lindeman, L. P.; Adams, J. Q. Carbon-13 Nuclear Magnetic Resonance Spectroscopy: Chemical Shifts for the Paraffins through C_9. *Anal. Chem.* **1971**, *43*, 1245–1252.
(9) Eggert, H.; Djerassi, C. J. Carbon-13 Nuclear Magnetic Resonance Spectra Of Acyclic Aliphatic Amines. *J. Am. Chem. Soc.* **1973**, *95*, 3710–3718.
(10) Roberts, J. D.; Weigert, F. J.; Kroschwitz, J. I.; Reich, H. J. Nuclear Magnetic Resonance Spectroscopy. Carbon-13 Chemical Shifts in Acyclic and Alicyclic Alcohols. *J. am. Chem. Soc.* **1970**, *92*, 1338–1347.
(11) Williamson, K. L.; Clutter, D. R.; Emch, R.; Alexander, M.; Burroughs, A. E.; Chus, C.; Bogel, M. E. Conformational Analysis by Nuclear Magnetic Resonance. Shift Reagent Studies on Acyclic Alcohols. ^{1}H and ^{13}C Spectra of the Six-Carbon Aliphatic Alcohols. *J. Am. Chem. Soc.* **1974**, *96*, 1471–1479.
(12) Quin, L. D.; Breen, J. J. Steric Effects in 31P NMR Spectra: "Gamma" Shielding in Aliphatic Phosphorus Compounds. *Org. Magn. Reson.* **1973**, *5*, 17–19.
(13) Ranc, L. M.; Jurs, P. C. Simulation of carbon-13 nuclear magnetic resonance spectra of quinolines and isoquinolines. *Anal. Chim. Acta* **1991**, *248*, 183–193.
(14) Bernassau, J. M.; Fetizon, M.; Maia, E. R. Prediction of Carbon-13 NMR Spectra. 1. Rigid Alkanes. *J. Phys. Chem.* **1986**, *90*, 6129–6134.
(15) Furst, A.; Pretsch, E. A computer program for the prediction of ^{13}C-NMR chemical shifts of organic compounds. *Anal. Chim. Acta* **1990**, *229*, 17–25.
(16) Clerc, J. T.; Sommerauer, H. A MiniComputer Program Based on Additivity Rules For the Estimation of ^{13}C-NMR Chemical Shifts. *Anal. Chim. Acta* **1977**, *95*, 33–40.
(17) Dubois, J. E.; Carabedian, M. Modelling of the Alkyl Environmental Effects on the ^{13}C Chemical Shift. *Org. Magn. Reson.* **1980**, *14*, 264–271.
(18) Lah, L.; Tusar, M.; Zupan, J. Simulation of carbon-13 spectra. *Tetrahedron Comput. Methodol.* **1989**, *2*, 5–15.
(19) Szalontai, C.; Recsey, Zs.; Csapo, Z. Use of ^{13}C-NMR Additivity Rules for the Ranking of Chemical Structures. *Anal. Chim. Acta* **1982**, *140*, 309–312.
(20) Pretsch, E.; Clerc, J. T.; Seibl, J.; Simon, W. *Tables of Spectral Data for Structural Elucidation of Organic Compounds*, 2nd ed.; Springer: Berlin, 1989.
(21) Brown, D. W. A Short Set of ^{13}C-NMR Correlation Tables. *J. Chem. Educ.* **1985**, *62*, 209–212.
(22) Bremser, W. HOSE-A Novel Substructure Code. *Anal. Chim. Acta* **1978**, *103*, 355–365.
(23) Munk, M. E.; Lind, R. J.; Clay, M. E. Computer Mediated Reduction of Spectral Properties to Molecular Structures: General Design and Structural Building Blocks. *Anal. Chim. Acta* **1986**, *184*, 1–19.
(24) Dubois, J. E.; Bonnet, J. C. The DARC Pluridata System: The ^{13}C-N.M.R. Data Bank. *Anal. Chim. Acta* **1979**, *112*, 245–252.
(25) Gray, N. A. B.; Nourse, J. G.; Crandell, C. W.; Smith, D. H.; Djerassi, C. Stereochemical Substructure Codes for ^{13}C Spectral Analysis. *Org. Magn. Reson.* **1981**, *15*, 375–389.
(26) Jurs, P. C.; Kowalski, B. R.; Isenhour, T. L. Computerised Learning Machines Applied to Chemical Problems. Molecular Formula Determination from Low Resolution Mass Spectra. *Anal. Chem.* **1969**, *41*, 21–27.
(27) Kowalski, B. R.; Jurs, P. C.; Isenhour, T. L.; Reilley, C. N. Computerised Learning Machines Applied to Chemical Problems: Interpretation of Infrared Spectrometry. *Anal. Chem.* **1969**, *41*, 1945–1949.
(28) Stonham, T. J.; Aleksander, I.; Camp, M.; Pike, W. T.; Shaw, M. A. Classification of Mass Spectra using Adaptive Digital Learning Networks. *Anal. Chem.* **1975**, *47*, 1817–1824.

(29) Ritter, G. L.; Woodruff, H. B. Dimensionality and the Number of Features in Learning Machine Classification Methods. *Anal. Chem.* **1977**, *49*, 2116–2118.
(30) Minsky, M. L.; Papert, S. *Perceptrons*; The MIT Press: Cambridge MA, 1969.
(31) Werbos, P. J. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University, 1974.
(32) Parker, D. B. *Learning Logic*; Invention Report S81-64, File 1; Office of Technology Licensing, Stanford University: Stanford, CA, 1982.
(33) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536.
(34) LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551.
(35) Sejnowski, T. J.; Rosenberg, C. R. Parallel Networks That Learn To Pronounce English Text. *Complex Syst.* **1987**, *1*, 145–168.
(36) Schoneburg, E. Stock Price Prediction Using Neural Networks. *Neurocomput. J.* **1990**, *2*, 17–27.
(37) Elrod, D. W.; Maggiora, G. M.; Trenary, R. G. Applications of Neural Networks in Chemistry: 1. Prediction of Electrophilic Aromatic Substitution Reactions. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 477–484.
(38) Aoyama, T.; Susuki, Y.; Ichikawa, H. Neural Networks Applied to Quantitative Structure–Activity Relationships. *J. Med. Chem.* **1990**, *33*, 905–908.
(39) Gezelter, D. J.; Freeman, R. Use of Neural Networks to Design Shaped Radiofrequency Pulses. *J. Magn. Reson.* **1990**, *90*, 397–404.
(40) Bos, A.; Bos, M.; Van der Linden, W. E. Artificial Neural Networks as a Tool for Soft-Modelling in Quantitative Analytical Chemistry: The Prediction of the Water Content of Cheese. *Anal. Chim. Acta* **1992**, *256*, 133–144.
(41) Brunak, S.; Engelbrecht, J.; Knudsen, S. Neural Network Detects Errors in the Assignment of mRNA Splice Sites. *Nucleic Acid Res.* **1990**, *18*, 4797–4801.
(42) Wythoff, B. J.; Levine, S. P.; Sterling, S. A. Spectral Peak Verification and Recognition Using a multilayered Neural Network. *Anal. Chem.* **1990**, *62*, 2702–2709.
(43) Long, J. R.; Gregoriou, V. G.; Gemperline, P. J. Spectroscopic Calibration and Quantization Using Artificial Neural Networks. *Anal. Chem.* **1990**, *62*, 1791–1797.
(44) Hinton, G. E. Connectionist Learning Procedures. *Artif. Intell.* **1989**, *40*, 185–234.
(45) Lippmann, R. P. An Introduction to Computing with Neural Nets. *IEEE ASSP Mag.* **1987** (Apr), 4–22.
(46) White, H. Neural Network Learning and Statistics. *AI Expert* **1989** (Dec), 48–52.
(47) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*; Rumelhart, D. E., McClelland, J. L., Eds.; The MIT Press: Cmabridge, MA, 1986; Vols. 1 and 2. *Neurocomputing*; Hecht-Nielsen, R., Ed.; Addison-Wesley: Reading, MA, 1989. *Introduction to the Theory of Neural Computation*; Hertz, J., Krogh, A., Palmer, R. G., Eds.; Addison-Wesley: Reading, MA, 1991.
(48) Zupan, J.; Gasteiger, J. Neural networks: A new method for solving chemical problems or just a passing phase? *Anal. Chem. Acta* **1991**, *248*, 1–30.
(49) Byers, W. A.; Perone, S. P. *k* Nearest Neighbor Rule in Weighting Measurements for Pattern Recognition. *Anal. Chem.* **1980**, *52*, 2173–2177, and references cited therein.
(50) Denker, J.; Schwartz, D.; Wittner, B.; Solla, S.; Howard, R.; Jackel, L.; Hopfield, J. Large Automatic Learning, Rule Extraction and Generalisation. *Complex Syst.* **1987**, *1*, 877–922.
(51) Quinlan, R. Induction of Descision Trees. *Mach. Learn.* **1986**, *1*, 81–106.
(52) Michalski, R.; Mozetic, I.; Hong, J.; Lavrac, N. The Multipurpose Incremental System AQ15 and Its Testing Application to Three Medical Domains. *Proceedings of AAAI-86*; Morgan Kaufman Publishers, Inc.: San Mateo, CA, 1986; pp 1041–1045.
(53) Mitchell, T. Version Spaces: A Candidate Elimination Approach to Rule Induction. *Int. Joint Conf. AI* **1977**, *5*, 305–310.
(54) Hinton, G. E. *Lectures on Neural Networks*, Section 3.21; University of St. Andrews 23rd Annual Open Lecture Course, Apr 18–19, 1991; University of St. Andrews: Scotland, U.K., 1991.
(55) Barrow, H. G. Neural Networks. *Proceedings of a one day tutorial course on Neural Networks*, University of Sussex, Great Britain, April, 1991; University of Sussex: Sussex, U.K., 1991; p 107.
(56) Refenes, A. N.; Azema-Barac, M.; Karoussos, S. A. Currency Exchange Rate Forcasting by Error Backpropagation. *Proc. of the Conference on System Sciences HICSS-25*, Kauai, Hawaii Jan 7–10 1992; IEEE Computer Society Press: Los Alemitos, CA, 1992.
(57) Kolmogorov, A. N. On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition. *Dokl. Akad. Nauk USSR* **1957**, *114*, 953–956.
(58) Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. *Math. Control, Signals, Syst.* **1989**, *2*, 303–314.
(59) Curry, Bo; Rumelhart, D. E. MSnet: A neural network which classifies mass spectra. *Tetrahedron Comput. Methodol.* **1990**, *3*, 213–237.
(60) Robb, E. W.; Munk, M. E. A Neural Network Approach to Infrared Spectrum Interpretation. *Mikrochim. Acta [Wien]* **1990**, *I*, 131–155.
(61) Munk, M. E.; Madison, M. S.; Robb, E. W. Neural Network Models for Infrared Spectrum Interpretation. *Mikrochim. Acta [Wien]* **1991**, *II*, 505–514.
(62) Davidge, R. Predicting Spectra Using Rule Induction and Neural Nets. *AISBQ Postgrad. AI Workshop* **1990**, 16–21.
(63) Mitchell, T. M.; Schwenzer, G. M. Applications of Artificial Intelligence for Chemical Inference XXV. A Computer Program for Automated Empirical ^{13}C NMR Rule Formation. *Org. Magn. Reson.* **1978**, *11*, 378–384.
(64) Kvasnicka, V. An Application of Neural Networks in Chemistry. Prediction of ^{13}C NMR Chemical Shifts. *J. Math. Chem.* **1991**, *6*, 63–76.
(65) Tebby, J. C. *Handbook of Phosphorus NMR Data*; CRC Press: Boca Raton, FL, 1991.
(66) Zupan, J.; Novic, M.; Bohanec, S.; Razinger, M.; Lah, L.; Tusar, M.; Kosir, I. Expert System For Solving Problems in Carbon-13 Nuclear Magnetic Resonance Spectroscopy. *Anal. Chim. Acta* **1987**, *200*, 333–345.
(67) IUPAC Physical Chemistry Division; Commission on Molecular Structure and Spectroscopy: Presentation of NMR data for Publication in Chemical Journals-B: Conventions Relating to Spectra from Nuclei Other than Proton; Recomendations 1975. *Pure Appl. Chem.* **1976**, *45*, 217–219.
(68) Wipke, W. T.; Dycott, T. M. Stereochemically Unique Naming Algorithm. *J. Am. Chem. Soc.* **1974**, *96*, 4834–4842.
(69) Shelly, C. A.; Munk, M. E. Computer Perception of Topological Symmetry. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 110–113.
(70) Herndon, W. C.; Leonard, J. E. Canonical Numbering, Stereochemical Descriptors, and Unique Linear Notations for Polyhedral Clusters. *Inorg. Chem.* **1983**, *22*, 554–557.
(71) Randic, M. On Unique Numbering of Atoms and Unique Codes for Molecular Graphs. *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 105–108.
(72) Balaban, A. T.; Mekenyan, O.; Bonchev, D. Unique Description of Chemical Structures Based on Hierarcically Ordered Extended Connectivities (HOC Procedures). III Topological Chemical and Stereochemical Coding of Molecular Structure. *J. Comput. Chem.* **1985**, *6*, 562–569.
(73) Moreau, G. A Topological Code for Molecular Structures. A Modified Morgan Algorithm. *Nouv. J. Chim.* **1980**, *4*, 17–22.
(74) Lederberg, J. Instrumentation Research Laboratory, Technical Report No 1140; Stanford University: Palo Alto, CA, 1966.
(75) Dubois, J. E. In *Computer Representation and Manipulation of Chemical Information*; Wipke, W. T., Heller, S. R., Feldman, R. J., Hyde, E., Eds.; Wiley: New York, 1974; Chapter 10, p 333.
(76) Jochum, C.; Gasteiger, J. Canonical Numbering and Constitutional Symmetry. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 113–117.
(77) Golender, V. E.; Drboglav, V. V.; Rosenblit, A. B. Graph Potentials Method and Its Application for Chemical Information Processing. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 196–204.
(78) Morgan, H. L. Generation of Unique Machine Description for Chemical Structures, a Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107–113.
(79) Leighton, R.; Wieland, A. *The Aspirin/MIGRANES Software Tools User's Manual*, Release V4.0; MITRE Washington Neural Network Group, Washington C^{3}I Division: McLean, VA, 1991.
(80) *Parallel Distributed Processing: A Handbook of Models, Programs and Exercises*; McClelland, J. L., Rumelhart, D. E., Eds.; The MIT Press: Cambridge, MA, 1986; Vol. 3, pp, 126–130.
(81) Plaut, D.; Nolan, S.; Hinton, G. Experiments on Learning by Back-Propagation; Techical Report CMU-CS-86-126; Department of Computer Science, Carnegie Mellon University: Pittsburgh, PA, 1986.
(82) Knuth, D. E. *The Art of Computer Programming: Seminumerical Algorithms*, 2nd ed.; Addison-Wesley: Reading, MA, 1981; Vol. 2.
(83) Refenes, A. N.; Alippi, C. Histological Image Understanding by Error BackPropagation. *Microprocess. Microprogramm.* **1991**, *32*, 437–446.