Journal of Thermal Analysis, Vol. 46 (1996) 833-851

Dedicated to Professor Bernhard Wunderlich on the occasion of his 65th birthday

ON THE USE OF COMPUTATIONAL NEURAL NETWORKS FOR THE PREDICTION OF POLYMER PROPERTIES

B. G. Sumpter and D. W. Noid

Chemical and Analytical Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6197, USA

Abstract

A general purpose computational paradigm using neural networks is shown to be capable of efficiently predicting properties of polymeric compounds based on the structure and composition of the monomeric repeat unit. Results are discussed for the prediction of the heat capacity, glass transition temperature, melting temperature, change in the heat capacity at the glass transition temperature, degradation temperature, tensile strength and modulus, ultimate elongation, and compressive strength for 11 different families of polymers. The accuracies of the predictions range from 1–13% average absolute error. The worst results were obtained for the mechanical properties (tensile strength and modulus: 13%, 7% elongation: 12%, and compressive strength: 8%) and the best results for the thermal properties (heat capacity, glass transition temperature, and melting point: <4%). A simple modification to the overall method is devised to better take into account the fact that the mechanical properties are experimentally determined with a fairly large range (due to variability in measurement procedures and especially the sample). This modification treats the bounds on the range for the mechanical properties as complex numbers (complex, modular neural networks) and leads to more rapid optimization with a smaller average error (reduced by 3%).

Keywords: complex backpropagation, computational neural networks, molecular structure, physical and mechanical properties, polymeric materials, quantitative structure-property relationships, statistical regression, unsupervised learning

Introduction

Professor Wunderlich continues to be at the forefront of polymer science research [1–4]. His intense pursuit of accurately defining and quantifying thermal properties of polymeric materials has set the standard for thermal analysis [5]. In addition, he has been active in using computational methods [6] to augment experimental research, an approach which has provided a substantial boost toward the general understanding of structure-property relationships and the fundamental importance of the microscopic dynamics to the observed macroscopic structure and properties.

0368–4466/96/ $ 5.00

John Wiley & Sons, Limited
Chichester

Professor Wunderlich would be among the first to agree that polymer science is of fundamental significance to science and technology. Polymers and polymer processing cuts across almost every sector of industry. From microelectronics to composites, from pharmaceuticals to petrofuels, from aerospace alloys to smart materials, our industrial base is built on our ability to invent and craft new materials. However, up to now, materials design and processing have been to a large extent empirical sciences: we are still unable to design new alloys and polymers to meet application specific requirements. Obviously, the ability to rapidly screen possible designs for specific applications, would provide a distinct advantage, potentially leading to significant advances in quantity and quality of materials products.

In this paper we present results obtained by using computational neural networks as tools for building reliable capabilities (an atomistic approach) in predicting materials properties. In the next section, the three types of neural networks used in this study are described followed by a discussion of the results in Sec. III and the conclusions in Sec. IV.

Computational scheme

Recently there have been a number of advances in the area of quantitative structure-property relationships (QSPR) that provide computational avenues capable of estimating certain properties based on fundamental structure [7–9]. The majority of these developments and studies are based on either fitting via standard statistical techniques [10] a set of descriptors based on group contributions or graph theory to known experimental data. However, the rather sparse set of data as well as the fact that there is significant variability in how data is reported and measured, can significantly degrade the ability to determine adequate parameters for the regression equations. The apparent ability of computational neural networks to adequately handle multimodal, noisy, and even sparse data [11] is perhaps essential toward improving predictive capabilities of standard QSPR methods [12].

Since the details of computational neural networks may be relatively new to the thermal analysis community, we briefly discuss fundamental concepts for the paradigms that we have employed in this study.

A. Computational neural networks

Computational neural networks (CNNs) are model-free estimators that have exceptional ability for performing multidimensional, nonlinear vector mappings. In general, CNNs have several essential constructs that define their operation: nodes (simple processing units), transfer functions (generally nonlinear and bounded functions), connection weights, and a learning algorithm.

One possible arrangement of the nodes is an architecture that describes a multilayer feedforward network: a set of nodes placed into two or more layers. An example of the basic architecture for a multilayer feedforward CNN is shown in Fig. 1. There is an input layer and an output layer, each consisting of at least one node. The nodes in the input layer do not perform any actual processing but serve only to distribute the input to the next layer. There are usually one or more hidden layers (layers of nodes between the input and output). The term feedforward means that the inputs to the nodes in each layer comes exclusively from the outputs of nodes in the previous layer, and the outputs from these nodes pass to nodes in the following layer. Each node in the network has a number of weighted connections to other individual nodes and an input signal is propagated through the system until it emerges as a network output. An optimization procedure that adjusts the weights connecting the nodes in order to minimize the difference between the output and the target (the desired result), is called the learning algorithm. Backpropagation [13] was the first practical method for training multilayer feedforward networks and is still the most popular learning algorithm.

Backpropagation

The backpropagation algorithm is an example of supervised learning (training with a teacher, that is, with known answers for representative examples). This algorithm adjusts the weights based on a gradient descent minimization of an error function (usually the sum of squared errors). A representation of a general architecture for a multilayer feedforward neural network trained with backpropagation was shown in Fig. 1. The goal is to 'teach' the network to associate specific output to each of several inputs. Having learned the fundamental relationship(s) between inputs and outputs, the neural network should then be able to produce reasonable output for unknown input (called generalization). The basic learning procedure can be summarized as follows:

(1) Initialize the node connection weights w_{ij} to some small random values.

(2) Input some data V_i^m and corresponding output values V_i^T, where m is the layer number, i is the node number, and T represents the target or desired output state.

(3) Propagate the initial signal forwards through the network using:

$$\mathrm{NET}_j^m = \Sigma w_{ij}^m V_i^{m-1} + \beta_j \quad \text{and}$$

$$V_j^m = F(\mathrm{NET}_j^m) \tag{1}$$

where w_{ij} is the connection weights between nodes i and j, V_i^{m-1} is the signal from node i and layer $m-1$, β is the threshold or bias value of the node, and F

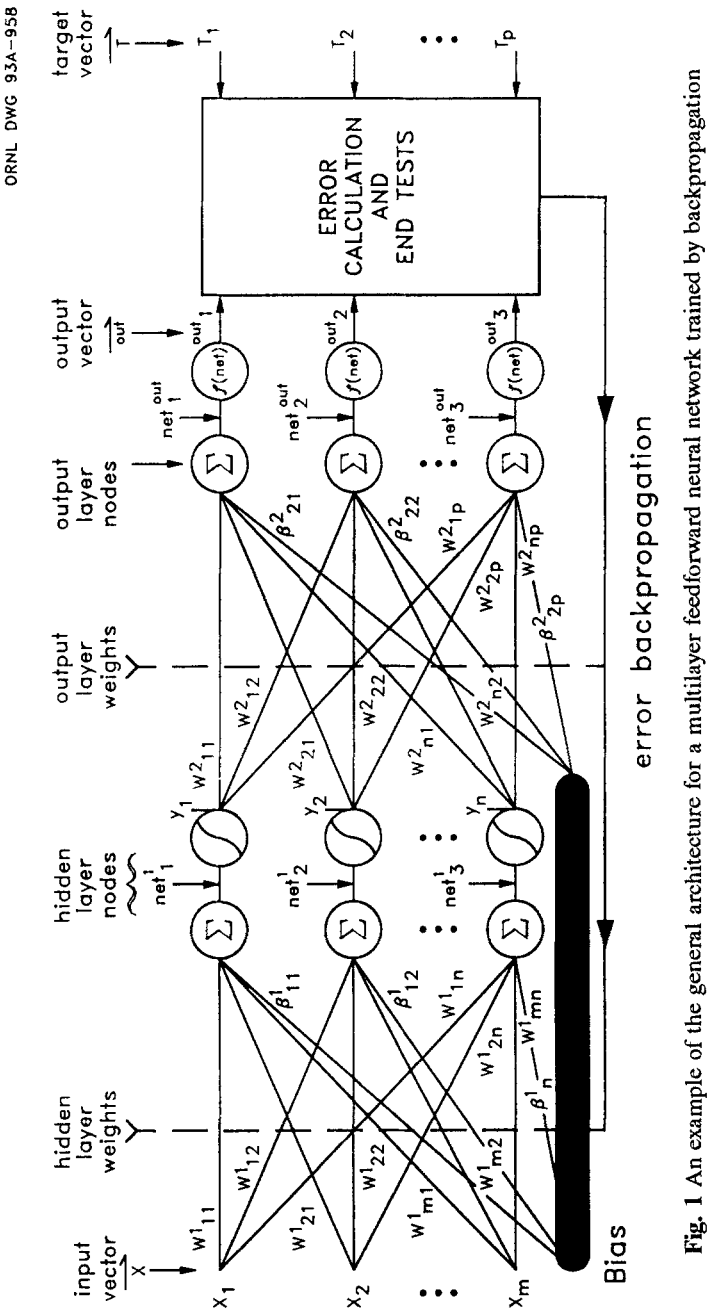

Fig. 1 An example of the general architecture for a multilayer feedforward neural network trained by backpropagation

is a transfer function usually taken as a sigmoid function, most commonly the logistic function:

$$F(\mathrm{NET}_j^m) = 1/[1 + \exp(-\mathrm{NET}_j^m)] \tag{2}$$

This function is continuous and varies monotonically from a lower bound of 0 to an upper bound of 1. It has a continuous derivative (Eq. (4)) which is a requirement of the standard backpropagation algorithm. The feedforward propagation (Eq. (1)) is continued for each i and m until the final outputs V_i^o have all been calculated.

(4) Compute the deltas (δ) for the output layer, defined as:

$$\delta_i^o = -\partial E/\mathrm{NET}_i^o = -(\partial E/\partial V_i^o)(\partial V_i^o/\partial \mathrm{NET}_i^o) =$$

$$= F'(\mathrm{NET}_i^o)[V_i^T - V_i^o] \tag{3}$$

where the error function $E = 1/2 \Sigma (V_i^T - V_i^o)^2$ and $F'(\mathrm{NET}_i^o)$ is the derivative of the transfer function with respect to the activation NET_i. For the logistic function the derivative is:

$$F'(\mathrm{NET}_i^o) = \partial F(\mathrm{NET}_i^o)/\partial \mathrm{NET}_i^o = F(\mathrm{NET}_i^o) \cdot [1 - F(\mathrm{NET}_i^o)] \tag{4}$$

where $F(\mathrm{NET}_i^o)$ is given by Eq. (2).

(5) Compute the deltas for the preceding layers by propagating the errors backward:

$$\delta_i^{m-1} = F'(\mathrm{NET}_i^{m-1})[\Sigma w_{ij}^m \delta_j^m] \tag{5}$$

for all $m = m, m-1, m-2, \ldots$, until it has been calculated for each layer.

(6) Using:

$$\Delta w_{ij}^m = \eta \delta_i^m V_j^{m-1} \tag{6}$$

where η is the learning rate, update the connection weights to:

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij} \tag{7}$$

(7) Return to step (2) and repeat for another input example.

This process (steps 1–7) is continued until the network output satisfies some ending criteria. In practice this type of iterative approach can often take a relatively large number of epochs (cycles through the whole data set) before a reasonable error is reached. Fortunately there are a number of methods which can help alleviate this pathology, although each modification comes with a number of new problems [14, 15]. For example, we have had good success with using

the Levenberg-Marquardt compromise to Newton's method for optimization problems [16]. This method basically interpolates between gradient descent and Gauss-Newton methods depending on the distance away from the minimum (gradient descent is used for minimizing the error for positions that are far away from a minimum and Gauss-Newton is used for those close to a minimum). This approach is quite practical since it is well-known that 2nd order or quasi-2nd order methods converge much faster than gradient descent close to a minimum (the error surface is quadratic: a well-defined Hessian matrix) but are slower when the error surface is not parabolic (far away from the minimum: the Hessian is not positive definite). Addition of a stochastic perturbation term to this method, much in the same spirit as Langevin dynamics, can also give improved performance (helps to escape from local minima in the hope of finding a global minima).

One final note, rather using the standard backpropagation or one of its may variants, it is sometimes prudent to carry out all calculations in the complex domain [17]. This type of approach is usually referred to as complex backpropagation and can actually lead to some surprising advantages. Training speed and reliability usually increase dramatically and generalization quality is almost always superior. The implementation is a minor modification of Eqs (1–7) above: there will be a real part (*Re*) and an imaginary part (*Im*) to the input, output, bias, transfer function and connection weights. It is important to recall that addition and multiplication in the complex domain are defined as: $(a,b)+(c,d)=(a+c), (b+d)$; $(a,b)^*(c,d)=(ac-bd,ad+bc)$. Otherwise the learning algorithm does not change: the derivatives (now with two parts, *Re* and *Im*) with respect to the weights are used to optimize the same error function (also with *Re* and *Im* parts). Complex domain neural networks are appropriate primarily when the data intrinsically occurs in pairs of numbers.

Kohonen self-organizing feature maps

An algorithm called Kohonen self-organizing feature maps, developed by Kohonen [18] is an example of unsupervised competitive learning. Competitive learning uses a winner-take-all environment in which the output nodes of a network compete for being the one that fires. The goal is to cluster or categorize the input data: similar inputs should map onto similar outputs. The architecture of a general networks is defined as illustrated in Fig. 2. There are n-dimensional input vectors, each fully connected to a grid on nodes by a n-dimensional weight vector w_{ij}. The grid of nodes have lateral connections and define the topology of the output space. The algorithm used to train the network utilizes an Euclidean distance measure to determine which node has a weight vector that is closer to the input vector. The one with the closest value is the winning node. The weight of this node is adjusted as well as a neighborhood of nearest nodes. Thus the grid (it can also be a linear array, hexagonal array, or other geometric figures) is organized into local neighborhoods that act as feature classifiers of the input data. Basically, the Kohonen algorithm moves the weight vectors so

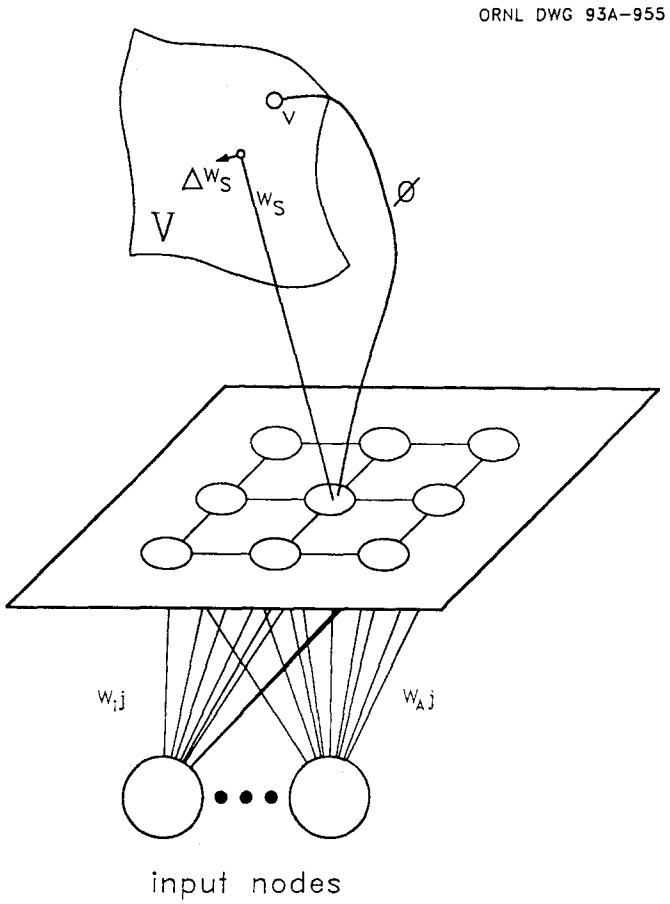

Fig. 2 An example of the general architecture of a Kohonen self-organizing feature map

that it more closely aligns with the input vectors. Since both input and weight vectors are normalized to a unit magnitude (a requirement), the vectors point to a position on a unit hypersphere (or circle for two dimensions). Using this algorithm creates a mapping of multidimensional information onto a layer of nodes that preserves the essential content of the information (depending on the dimensionality of the input, it can also serve as a data compression method).

Mathematically the algorithm represents a so-called Markov process (a set of states and a set of transition probabilities between states that determine a stochastic process which produces as sequence of states). The general algorithm can be defined as:

(1) Initialize the weights w_{ij} for defined network (usually a grid as shown in Fig. 2 or an array of nodes) to some small values: chose a subset of random numbers from a uniform random distribution between 0 and 1 that is centered at 0.5, 0.5 with a small width (0.1 for example).

(2) Present an input vector, $x(t)$.

(3) Calculate the Euclidian distances between the input and all of the weights and select the node with the smallest distance, r_i^*.

(4) Update the weights for the winning node and its neighbors, defined by the neighbourhood size $N_j(t)$:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\, N_j(t)\, [r_i(t) - w_{ij}(t)] \tag{8}$$

where η is the learning rate and decreases with time so that the weight adaption is reduced to zero and

$$N_j(t) = \exp(-\,|r_i - r_i^*\,|^2 / 2\sigma^2) \tag{9}$$

The neighborhood function, $N_j(t)$, is shown here as a Gaussian with a width parameter of σ and a center determined by the winning node labeled r_i^*. The width parameter is reduced during training to a final value of 1. One method for reducing the learning rate η and the Gaussian width parameter σ is $\sigma(t+1)=[0.1/\sigma(t)]^{(1/K)}$, $\eta(t+1)=[0.1/\eta(t)]^{(1/K)}$, where K is the total number of anticipated cycles.

(5) Return to 2 and repeat until all of the examples have been examined.

(6) Go to 2 if convergence has not been achieved.

Adaptive resonance theory

Adaptive resonance neural networks are meant to model a self-organizing pattern recognition network capable of processing nonstationary data. There are a series of these networks proposed and develop by S. Grossberg and G. Carpenter [19–21] (ART1, ART2, ART3, fuzzy ART, ART2a, etc) that have the ability to switch modes between plastic (internal parameters can be modified) and stable (a fixed classification set), without loosing any result that had been previously learned (encoded). The architecture of an adaptive resonance network is one in which two layers are fully connected with adaptive weights between them that have both feedforward and feedback connections (Fig. 3). The layer at the bottom is the input layer (F1 field) and the upper layer is the storage layer (F2 field). Each training pattern (ART is an unsupervised network) is presented to the network and causes an activity in the input layer. A transmission to every node in the top layer (bottom-up pattern) occurs, and the pattern is modified through the connections in the upper layer that stimulates a response pattern (top-down pattern). Since the two layers are fully connected, the top-down pattern is presented back to the input layer. If the pattern of activity excited in the input layer nodes by the top-down input is a close match to the pattern excited in the input layer by external input, the system is said to be in adaptive resonance. Eventually, either the pattern is placed in an existing category or learned as the first example of a new category.

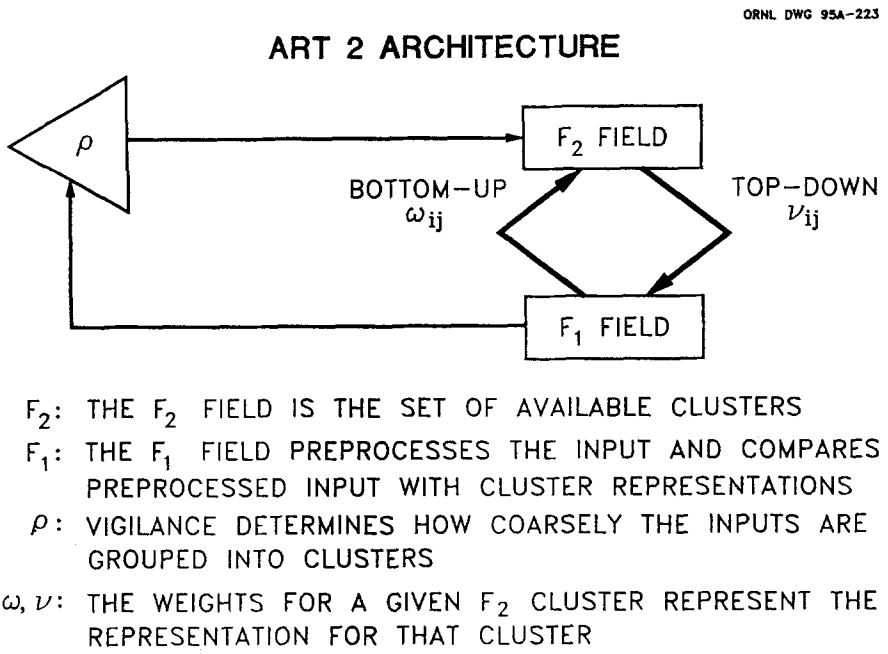

Fig. 3 Schematic of the architecture and learning algorithm for an ART2 neural network

The ART algorithm is very good at cluster discovery. It has the novel ability to perform controlled discovery of clusters and can accommodate new clusters without affecting the storage or recall capabilities of the clusters already learned. The general algorithm is:

(1) Initialize the weights (M output nodes and N input nodes): top-down and bottom-up. These weights define the exemplar specified by the output node.

(2) Set the vigilance threshold ρ: this determines how close an input has to be to correctly match a stored exemplar.

(3) Apply an input I.

(4) Compute best matching cluster j (w_{ij} are bottom-up weights).

$$\mu_j^* = \max_j[\mu_j], \text{ where } \mu_j = \Sigma\, w_{ij}\,(t)\, I_i \text{ for } 0 \leq j \leq M-1 \tag{10}$$

(5) Perform the similarity test for I and cluster j.

$$\Sigma\, v_{ij} I_i \,/\, ||\, I\, || > \rho, \tag{11}$$

where $||\, I\, || = \Sigma I_i$ and v_{ij} are top-down weights.

(6) If vigilance is passed go to 8 else to 7.

(7) Disable the best match by setting the output of the best match node to 0, go to 4.

(8) Update the weight matrices for index j:

$$w_{ij}(t+1) = v_{ij}(t)I_i \,/[0.5 + \Sigma v_{ij}(t)I_i], \quad v_{ij}(t+1) = \Sigma I_i v_{ij}(t) \tag{12}$$

enable any disabled nodes, then go to step 3.

The training algorithm for ART has a different learning philosophy than other neural network paradigms. The learning is optimized to enable the network to re-enter the training mode at any time in order to incorporate new training data. The network will continue to add new information, until it uses all of the available memory while refining stored information within it as new information is presented. The algorithm has been proven to be stable (no convergence problems as in backpropagation). Although there are questions concerning its stability to noisy input conditions (especially for ART1), latter versions have been substantially improved (Fuzzy ART) [20]. In addition, Fuzzy ARTMAP [21], an incremental supervised version of Fuzzy ART, is capable of performing very efficient and accurate classifications.

B. Numerical representation of molecular structure

A given molecular structure for a repeat unit of a particular polymer, Kapton as shown in Fig. 4, is transformed into a set of numerical descriptors using methods founded in graph theory and/or chemical nomenclature. Initially a large number of structural descriptors useful for characterizing molecular structure were examined in order to quantify which were the most optimal. These included the standard connectivity matrices/indices, Wiener numbers, path numbers, graph polynomials, weighted paths, and various combinations (over 100 different types) [22–25]. To facilitate the determination of the best representation of the fundamental structure of polymers, we have used unsupervised neural networks. In this approach, the neural network formulates clusters of data that is statistically similar. If the data for a set of 11 different families of polymers (polyolefins, polyoxides, polhhalo-olefins, polyvinyls, polystyrenes, polyacrylates, polymethacrylates, polyacrylics, polydienes, polyesters, polycarbonates) is used as input to the self-organizing feature map or adaptive resonance theory networks, the final 'converged' result should indicate how many structurally different families are present. For the purpose of accurately predicting polymer properties, the use of the descriptors defined by a combination of molecular connectivity indices [25], chemical composition, and standard IUPAC nomenclature for naming compounds, was determined to provide optimal results (ART2 found 11 different clusters). Alternatively, the same descriptors could have each been used to train a supervised neural network by

simply testing each possible combination (a heuristic-brute force method). Genetic algorithms [26, 27] or other combinatorial optimization methods can also be effectively used to sort out the best representation. In this case the basis for selecting a particular set of descriptors depends on optimizing the performance of a neural network to predict the properties (if the particular combination of descriptors is a bad representation, the neural network should not be able to form any reasonable correlations to the properties: as in the heuristic-brute force method). We tried all of these approaches and found the same answer, however the unsupervised methods were by far the easiest and fastest method to employ. The optimal descriptors discovered are based on topological indices obtained by using graph theory, and reflect both topological connectivity, information on the electronic structure, chemical composition, and bulk orientation. In addition, the overall scheme takes into account properties that depend on the amount of material present, called extrinsic and those which do not, called intrinsic [7].

The appropriate 'molecular' indices are used as input to a set of neural networks which then utilize this information to correlate the fundamental structure to a set of polymer properties (Fig. 4). Figure 4 features the multiple 'local expert' or modular approach to neural network computing. That is each property of interest is correlated to the structural representation by using individual neu-

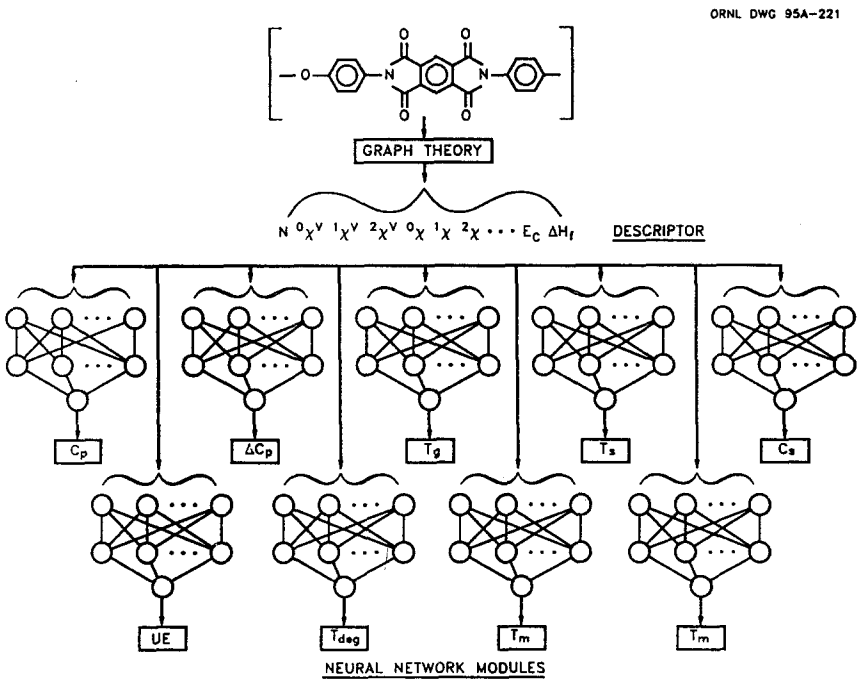

Fig. 4 Schematic of the modular neural network approach to predicting polymer properties

ral networks (9 properties ==>9 neural networks; the local experts). This actually allows a better prediction of the properties since each network is optimized on only one output variable (a divide and conquer strategy), plus it allows for the fact that the same amount of data for each property is not generally available. A global prediction, is of course possible (all properties using one network), and has been presented for some polymer properties in our previous work [12].

Results and discussion

Several computational experiments were conducted. First, an attempt was made to use a CNN to correlate the molecular indices to the parameters of the Tarasov function for heat capacity. If this correlation could be made, then the prediction of heat capacity at any temperature could be made using the Tarasov equation [28] which is already set up as part of the ATHAS data bank [29]. This would be a significant improvement since there are a large number of polymers for which the information required to formulate the parameters is not readily available and is generally obtained by the trial and error inversion of heat capacities to the approximate vibrational spectrum (a tedious and perhaps inaccurate procedure). Previous work demonstrated that the Tarasov function could be inverted using a neural network to obtain the two parameters (Θ_1,Θ_3) for the skeletal vibrations of 36 macromolecules [30]. However this approach explicitly assumes that the heat capacity of polymers is indeed described by the Tarasov function. Wunderlich and co-workers have persuasively demonstrated that the Tarasov function is a reasonable model of the temperature dependence of the heat capacity over a certain temperature range, above which, some deviation is found (presumably due to the anharmonic contribution of conformational disorder). Thus it could be argued that obtaining the best parameters for the Tarasov function is a logical approach toward generalizing the computation of heat capacities for polymers. In any case, this approach requires the existence or measurement of heat capacity of the skeletal vibration (which implies that there is linear separability of the heat capacity into the skeletal and group vibrational contributions) as a function of temperature. A more atomistic approach would be to predict the values of Θ_1,Θ_3 directly from the composition and structure of the polymer. It is clear that structure (electronic and geometric) and the atomic composition of molecules dictate the behavior of the vibrational modes. Thus it is reasonable to expect that, if the vibrational spectrum can be predicted based on structure/composition information, the parameters of the Tarasov function could also be predicted equally well (of course this assumes that Θ_1,Θ_3, which define 1 and 3-dimensional Debye functions, are representative of the vibrational spectrum). A CNN was presented structural information on a series of 53 different polymers for which optimal values of Θ_1,Θ_3 were available

(from the ATHAS data bank). Backpropagation using the Levenberg-Marquardt method for optimization was used to train the network architecture (number of hidden layer nodes) which was automatically determined using complexity regulation. The training was terminated based on cross-validation (minimum obtainable error as monitored on a test set not the training set). In addition, an ensemble average of 50 different training and test sets were obtained using the bootstrap resampling technique (random sampling with replacement) [31]. This procedure nearly or completely eliminates any accidental bias used in selecting the division of the data into training and testing sets and in judgeing the overall performance of the neural networks' prediction capabilities. The resulting network gave an average prediction error of 9% for Θ_1 with a standard deviation of 50° and 14% with a standard deviation of 13° for Θ_3, somewhat disappointing considering the success of previous neural network studies. The best explanation that we can give for this behaviour is that there simply isn't any strong correlation possible between fundamental structure (electronic and geometric) and composition to the parameters of the Tarasov equation. Of course this statement assumes that we have accurately represented the structure and that a neural network is capable of forming nonlinear correlations. The latter condition is well-known to be true: neural networks are universal approximates (they can approximate any function). The representation that we have used has been well tested for the prediction of numerous properties for polymeric compounds (as is shown below) and was chosen using unsupervised neural networks (an unbiased technique). In addition, to be certain that there wasn't something subtle in the representation, we tested a number of other graph theoretic representations which prove reasonable descriptions for structure (including representations of the monomer, dimer, and trimer units of the parent polymer: giving a total of 296 different representations). No improvement in accuracy was found (of course this is of no suprise since the representation was optimized, as described in section IIB, based on a similar set of possible descriptors).

The approach that we have found the most profitable is to directly predict polymer properties (thermal and mechanical) from the structural representation. Table 1 illustrates the accuracies of this neural network approach and shows a comparison to that obtained from some statistical regression techniques [linear, polynomial, and spline partial least squares regression (PLS, PPLS, SPLS) [32–34], locally weighted [35], ridge [36], and kernel regression [37] (LWR, RR, KR)]. In all cases, cross-validation was used to determine the optimal parameters for the various forms of regression (the number of latent variables, the number of points to use in the locally weighted regression, the best values for the regression parameter theta in ridge regression, the degree of the polynomial used in polynomial and spline partial least squares, the width of

the local basis functions used in KR). One note should be called to attention. Most of these statistical methods all assume that the data being examined can be fit to an underlying model: for PLS and RR this model is linear, for PPLS it is polynomial [the degree is specified] and LWR is linear or quadratic. Kernel regression is general in theory since it uses local basis sets (usually Gaussian kernels) optimized to determine the probability density function directly from the data.

The error measures given in Table 1 are defined as follows: $E_{\text{residual}} = |T-P|$, where T is the known answer and P is the prediction; $<\%\text{error}> = \Sigma E_{\text{residual}}/T \cdot 100$, correlation $= [\Sigma(T_i - <T>)(P_i - <P>)]/[\sqrt{(T_i - <T>)^2}\sqrt{(O_i - <O>)^2}]$, std(residual) $= \sqrt{[<E^2_{\text{residual}}> - <E_{\text{residual}}>^2/N - 1]}$. Table 1 indicates that the neural network prediction of the thermal and mechanical properties achieves a better accuracy than any of the statistical methods. Since the structure-property relationships are most likely nonlinear in multiple variables, it is clear that the linear models of PLS and RR will have less of a chance of accurately predicting new results. While PPLS and LWR share the relative computational simplicity of PLS, it is possible that as a soft-modeling technique, their use of simple polynomial expansions to describe complex, nonlinear response surfaces, may still not be optimal. A more recent version of nonlinear PLS uses a splined inner re-

Table 1 Predictions of various thermal and mechanical properties for polymeric materials using neural networks and statistical regression techniques*

Predicting C_p [Property range: 38–850 J mol^{-1} K^{-1}; 170 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	1.9	0.997	5 J mol^{-1} K^{-1}
PLS	6.0	0.980	11 J mol^{-1} K^{-1}
LWR	2.5	0.989	10 J mol^{-1} K^{-1}
RR	6.1	0.900	11 J mol^{-1} K^{-1}
PPLS	4.9	0.987	10 J mol^{-1} K^{-1}
KR	4.9	0.987	10 J mol^{-1} K^{-1}

Predicting ΔC_p [Property range: 9–177 J mol^{-1} K^{-1}; 56 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	3.7	0.98	3.0 J mol^{-1} K^{-1}
PLS	12	0.92	4.3 J mol^{-1} K^{-1}
LWR	6.5	0.92	6.0 J mol^{-1} K^{-1}
RR	12	0.93	4.4 J mol^{-1} K^{-1}
PPLS	10	0.94	4.0 J mol^{-1} K^{-1}
KR	11	0.62	22.0 J mol^{-1} K^{-1}

Predicting T_m [Property range: 230–66 K; 56 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	4	0.98	21°
PLS	9	0.95	26°
LWR	7	0.96	24°
RR	9	0.95	26°
PPLS	11	0.94	30°
KR	5	0.95	40°

Predicting T_g [Property range: 130–68 K; 320 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	6	0.99	7°
PLS	14	0.90	30°
LWR	12	0.80	50°
RR	14	0.90	30°
PPLS	11	0.93	27°
NPLS	11	0.82	15°

Predicting T_{dg} [Property range: 321–47 K; 24 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	9	0.93	9°
PLS	41	0.30	21°
LWR	16	0.69	19°
RR	40	0.23	25°
PPLS	19	0.73	14°
KR	11	0.82	15°

Predicting Tensile Strength [Property range: 100–900 kg cm^{-2}; 24 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	13	0.90	40 kg cm^{-2}
PLS	34	0.28	116 kg cm^{-2}
LWR	32	0.45	205 kg cm^{-2}
RR	33	0.27	112 kg cm^{-2}
PPLS	34	0.31	86 kg cm^{-2}
KR	23	0.51	113 kg cm^{-2}

Predicting Tensile Modulus [Property range: 1800–36000 kg cm^{-2}; 24 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	6.8	0.95	951 kg cm^{-2}
PLS	38	0.18	6000 kg cm^{-2}
LWR	25	0.37	5716 kg cm^{-2}
RR	36	0.14	5745 kg cm^{-2}
PPLS	26	0.48	3709 kg cm^{-2}
KR	19	0.48	5018 kg cm^{-2}

Predicting Compressive Strength [Property range: 60–2000 kg cm^{-2}; 24 examples]

Method	< %Error >	Correlation	Std(residual)
FNN	8	0.92	50 kg cm^{-2}
PLS	34	0.32	182 kg cm^{-2}
LWR	23	0.58	198 kg cm^{-2}
RR	31	0.31	177 kg cm^{-2}
PPLS	38	0.50	151 kg cm^{-2}
KR	17	0.56	149 kg cm^{-2}

Predicting Ultimate Elongation [Property range: 2–750%; 24 examples]

Method	< %Error >	Ccorrelation	Std(residual)
FNN	12	0.90	15%
PLS	27	0.63	38%
LWR	17	0.76	33%
RR	27	0.63	39%
PPLS	26	0.66	29%
KR	19	0.62	51%

* Definition of the abbreviations used in the table:
FNN: Feedforward neural networks trained with backpropagation
PLS: Linear partial least squares regression
LWR: Locally weighted regression (a nonlinear regression technique)
RR: Ridge regression (linear technique)
PPLS: Polynomial partial least squares regression
KR: Kernel Regression

lation (with user specified knots), which in principle should be able to model more complex relationships. We have also used the spline-PLS (SPLS) on the problems in Table 1. No notable improvement over PPLS was obtained (not shown in the Table 1). Obviously either the amount of data used was not suffi-

cient for the statistical techniques to determine an optimal regression vector or the correlation between the input and output is more complicated than any linear or simple polynomial functions can to accurately describe (KR however should be able to fit any relationship). The power of the neural network approach is that it is model free. The best set of 'functions' is determined automatically from the data, independent of the complexity of this function. Furthermore, it appears that the very things (such as sparse data, outliers, multimodal distributions, fuzzyness) that can cause some of the traditional methods to fail or significantly degrade, are not such an issue for neural networks (at least in this study). One issue of course is the computational time and difficulty in successfully using the various methods. All methods were computational efficient for this problem, requiring only a few CPU seconds, or at most, minutes on an IBM RISC 6000/580 to obtain the results (this includes training for the neural network). It is difficult to be unbiased in the degree of difficulty in using the various methods. So we won't comment on this other than to say that cross-validation and bootstrap resampling [31] was used for each method to determine training/test sets of data, adjustable parameters, and end conditions. All of these methods provide useful soft-modeling capabilities and should in general be used in a complimentary manner (the statistical techniques can provide information of how the problem is solved). In any case, the results obtained from the neural network are the most satisfactory and clearly demonstrate that predictions of polymer properties are possible. Although no attempt was made to predict the temperature dependence of the properties, it is clear that neural networks are capable of formulating such correlations based on previous work for heat capacity predictions [37].

An improvement is possible in the prediction of mechanical properties. Since the measurement of tensile and compressive strengths and moduli is strongly dependent on the sample itself (as well as other variables such as humidity etc), data is usually reported within a range (lower and upper bounds). Assigning a fixed value for the tensile strength forces the neural network to determine specific correlations to that value. A better approach, not considering adaptive fuzzy systems (which actually works very well), would be to give the information on the bounds of the individual properties to the neural network. In this manner, even though the structural representation is constant, the correlations that are formulated by the neural network should better take into account the variability of the data. One way to present data that have two variables per observation is to use complex domain neural networks. Thus, a neural network is simply presented with the same structure representation, but the training and the output are carried out using complex numbers. The results for the prediction of mechanical properties showed an notable improvement: average absolute error decreased by 3%. Thus the overall approach to predicting properties of polymeric materials based on structural representations appears to work very well for many different properties.

Conclusions

A computational methodology has been discussed for making accurate predictions of polymer properties based on their molecular structure. Results of this paper demonstrate that the accuracy is reasonable for a number of thermal and mechanical properties. Extension to more specific problems such as predicting heat capacity as a function of temperature should be relatively straight forward, thus representing a powerful method in the refinement and extension of data from thermal mechanical analysis.

* * *

This research was sponsored by the Division of Materials Sciences, Office of Basic Energy Sciences, U.S. Department of Energy, under Contract No. DE-AC05-84R21400 with Lockheed Martin Energy Systems, Inc. We would like to express our gratitude for the continued collaboration, support, and interest of Prof. Wunderlich in our research. We would also like to thank participants of the 1st DOE Workshop on Applications of Neural Networks in Materials Sciences for useful discussion on materials properties and neural networks.

References

1 B. Wunderlich, Macromolecular Physics, Crystal Structure, Morphology, Defects, Academic Press, New York 1973.
2 B. Wunderlich, Macromolecular Physics, Crystal Nucleation, Growth, Annealing, Academic Press, New York 1976.
3 B. Wunderlich, Macromolecular Physics, Crystal Melting, Academic Press, New York 1980.
4 B. Wunderlich, M. Möller J. Grebowicz, and H. Baur, Conformational Motion and Disorder in Low and High Molecular Mass Crystals, Springer, New York 1988.
5 B. Wunderlich, Thermal Analysis, Academic Press, New York 1990.
6 B. G. Sumpter, D. W. Niod, G. L. Liang, and B. Wunderlich, Adv. Poly. Sci., 116 (1994) 27.
7 J. Bicerano, Prediction of Polymer Properties, Marcel Dekker, Inc., New York 1993.
8 D. W. van Krevelen, Properties of Polymers, 3rd edition, Elsevier, Amsterdam 1990.
9 D. W. van Krevelen, in Computational Modeling of Polymers, edited by J. Bicerano Marcel Dekker, New York 1992.
10 M. Stone and P. Jonathan, J. Chemometrics, 8 (1994) 1.
11 B. G. Sumpter, C. Getino, and D. W. Noid, Annu. Rev. Phys. Chem., 45 (1994) 439.
12 B. G. Sumpter and D. W. Noid, Makromol. Chem., Theory Simul., 3 (1994) 363.
13 P. J. Werbos, The Roots of Backpropagation, From Ordered Derivaties to Neural Networks and Political Forecasting, Wiley, New York 1994.
14 T. Masters, Practical Neural Network Recipes in C++, Academic Press, San Diego, CA 1993.
15 J. Hertz, A. Krogh, and R. G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA 1991.
16 M. F. Møller, Neural Networks, 6 (1993) 525; M. T. Hagan and M. B. Menhaj, IEEE Trans. Neural Networks, 5 (1994) 989.
17 T. Masters, Signal and Image Processing with Neural Networks, Wiley, New York, 1994.
18 T. Kohonen, Biological Cybernetics, 43 (1982) 59; T. Kohonen, Self-Organization and Associative Memory, 3rd edition, Springer Verlag, Berlin 1989.

19 G. A. Carpenter, S. Grossberg, Pattern Recognition by Self-Organizing Neural Networks, MIT Press, Cambridge 1991; G. A. Carpenter, S. Grossberg, and J. H. Reynolds, Neural Networks, 4 (1991) 565.
20 G. A. Carpenter, S. Grossberg, and D. B. Rosen, Neural Networks, 4 (1991) 759.
21 G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, IEEE Trans. Neural Networks, 3 (1992) 698.
22 M. Gordon and W. B. Temple, from Chemical Applications of Graph Theory, edited by A. T. Balaban, Academic Press, New York 1976.
23 M. Randic' in, Concepts and Spplications of Molecular Similarity, edited by M. A. Johnson and G. M. Maggiora, John Wiley & Sons, New York 1990, chapter 5.
24 N. Trinajstic', Chemical Graph Theory, (CRC Press, Boca Raton, Fl, 1992) second edition.
25 L. B. Kier and L. H. Hall, Molecular Connectivity in Structure-Activity Analysis, RSP Press, New York 1986.
26 D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, New York: Addison-Wesley 1989.
27 L. Davis, Handbook of Genetic Algorithms, New York: Van Nostrand Reinhold 1991.
28 V. V. Tarasov, Zh. Fis. Khim., 24 (1950) 111; 27 (1953) 1430; 39 (1965) 2077 and Dokl. Acad. Nauk SSSR, 100 (155) 307.
29 The ATHAS Data Bank, see for example.
30 D. W. Noid, M. Varma-Nair, B. Wunderlich, and J. A. Darsey, J. Thermal Anal., 37 (1991) 2295.
31 B. Efron, The Jackknife, the Bootstrap and Other resampling Plans, SIAM, Philadelphia, PA 1982.
32 P. Geladi and B. R. Kowalski, Anal. Chim. Acta., 185 (1986) 1.
33 S. Wold, N. Kettaneh-Wold, and B. Skagerberg, Chemom. Int. Lab. Syst., 7 (1989) 53.
34 S. Wold, Chemom. Int. Labs. Syst., 14 (1992) 71.
35 T. Naes, T. Isaksson, and B. R. Kowalski, Anal. Chem., 62 (1990) 664.
36 A. E. Hoerl, Technometrics, 12 (1970) 55; A. E. Hoerl, Chemical Engineering Progress, 58 (1962) 54.
37 G. A. F. Seber, and C. J. Wild, Nonlinear Regression, Wiley, New York 1989.
38 J. A. Darsey, D. W. Noid, and B. R. Upadhyaya, Chem. Phys. Lett., 177 (1991) 189.