Polymer property prediction and optimization using neural networks

N. K. Roy
Department of Computer Science and Center for Simulational Physics (Department of Physics and Astronomy), University of Georgia 30602, USA.

W. D. Potter
Department of Computer Science, University of Georgia 30602, USA.

D. P. Landau
Center for Simulational Physics (Department of Physics and Astronomy), University of Georgia 30602, USA.

ABSTRACT

Prediction and optimization of polymer properties is a complex and highly non-linear problem with no easy method to accurately predict polymer properties directly and accurately. The problem is even more complicated in high molecular weight polymers like engineering plastics which have the greatest use in industry. The effects of modifying a mer (polymer repeat unit) on the polymerization and the resulting polymer properties is not as easy a problem to investigate experimentally given the large number of possibilities. This severely curtails the design of new polymers with specific end use properties. Another aspect in the development of useful materials is the use of polymer blending. Here again predicting miscibility or compatibility together with resulting blend properties is difficult, but imperative in order to get a useful product. Since miscibility can be controlled using low molecular weight compatibilizers the problem for experimentalists to find the correct working combination is further exacerbated. In this paper we show how properties of modified mers can be predicted using Neural Networks. In an earlier paper[1] we have given an overall approach to polymer blend design in which the use of Neural Networks was but one stage in the process that also made use of genetic algorithms and Markov chains to accurately predict blend miscibility. Here we present in great depth the first step which is the prediction and optimization of modified polymer properties using Neural Networks. We utilize a large database of polymer properties and employ a wide variety of networks ranging from backpropagation networks to unsupervised self-associating types. Based on extensive training and comparisons we are able to select particular networks that can be used to predict accurately specific polymer properties. We are able to classify the networks in the design into groups that range from those that provide quick training to those that provide excellent generalization. We are also able to show how the extensive databases available on polymers can be easily used to accurately predict and optimize polymer properties.

Keywords: Neural networks, backpropagation, self-associating, learning rate, momentum, initial weights, polymers, main chain and side chains, pendant groups, steric factors, potentials, affinity, glass-transition, polymerization, modulus.

1. INTRODUCTION

Artificial neural networks are model-free estimators that have exceptional ability for performing multi-dimensional, non-linear vector mappings[2]. The most commonly and widely used network is the standard backpropagation network in which every layer is linked or connected to the immediately previous layer. Mathematically is has been shown that this type of network with at most two hidden layers[3] can solve any non-linear problem provided there are sufficient numbers of hidden nodes. Most areas in polymer modeling require the handling of highly non-linear problems in which the components are almost never linearly separable. The relationships between the parameters being modeled and the actual behavior of these variables in the real world must be correlated as precisely as possible. However in most cases this is not possible and several approximations and simplifications are almost invariably made at various stages. Also when dealing with sparse, noisy or incomplete data conventional methods (decision theoretic/statistical and syntactic[4]) almost certainly fail. In addition, conventional methods lack generalization, fail to incorporate statistical and systematic fluctuations, and in most cases are limited to finite state spaces. Thus in most cases the important correlations between the model developed and the real properties are lost or not captured correctly.

High molecular weight polymer systems represent one of the most complex classes of materials and are very difficult to model. Besides being highly non-linear systems, there are a large number of parameters that need to be accurately defined if such systems are to be properly characterized. It is no wonder that conventional methods can only just begin to model simple polymer systems such as linear high polymers, copolymers, and their blends, or low molecular weight branched polymers and copolymers, but cannot handle higher order complex systems like high molecular weight binary and ternary blends, polymer dispersed liquid crystals, ionomers and interpenetrating networks. However we have shown that these factors are easily overcome[1] in a classic example, that of polymer blend design. In this paper we discuss modeling polymers using the first of the techniques we have used, neural networks, in much greater depth and detail.

2. DATA SET AND NEURAL NETWORKS

Since design of a new mer that can be effectively polymerized and that will subsequently yield a stable and useful product has reached a limit we restrict our studies to modified polymers in which we consider replacing pendant side groups in the side or main chain of the polymer. This is both experimentally more feasible to achieve, and can exploit the vast information on existing polymers. One or more of the resulting polymers now can be used as parent polymers in studying miscibility and other properties in polymer blend systems.

Of the wide variety of polymers available probably the most important are the engineering polymers[5], and we concentrate on their properties (mechanical, thermal, magnetic, optical, electrical, environmental and deteriorative) and the relationships with their structures (microscopic, mesoscopic and macroscopic). The prediction of polymer properties from just the structure of the monomer is complicated. However trained neural networks given optimized input data do an excellent job of characterizing a new modified polymer that can then be easily synthesized. Not only can we predict a whole range of properties using this technique, it is even possible to investigate the criteria for successful polymerization and stability like the heat of polymerization[6].

2.1 Polymer Database

The polymer database used was selected carefully so as to include all possible types of polymers available. Within each category several different types of polymers were included to provide a comprehensive set of data for each class. This is given in TABLE I. The number in

1.Main-chain Acyclic Carbon Polymers
- Poly(dienes) (4)
- Poly(alkenes) (7)
- Poly(acrylic acid) (11)
- Poly(acrylic acid ester) (9)
- Poly(acrylamides) (2)
- Poly(methacrylic acid) (4)
- Poly(methacrylic acid ester) (4)
- Poly(methacrylamides) (2)
- Poly(vinyl ether) (2)
- Poly(vinyl thioether) (1)
- Poly(vinyl alcohol) (5)
- Poly(vinyl ketones) (2)
- Poly(vinyl halides) (5)
- Poly(vinyl nitriles) (5)
- Poly(vinyl esters) (25)
- Poly(styrenes) (48)

2.Main-chain Carbocyclic Polymers
- Poly(phenylenes) (19)

3.Main-chain Acyclic Hetroatom Polymers

3.1)Main-chain -C-O-C Polymer
- Poly(oxides) (22)
- Poly(carbonates) (46)
- Poly((esters) (70)
- Poly(anhydirdes) (1)
- Poly(urethanes) (32)

3.2)Main-chain O-Heteroatom Polymers
- Poly((sulphonates) (1)
- Poly(siloxanes) (15)

3.3)Main-chain -C-(S)-C- and -C-S-N- Polymers
- Poly(sulphides) (8)
- Poly((thioesters) (1)
- Poly(sulphones) (2)
- Poly(sulphonamides) (2)

3.4)Main-chain -C-N-C Polymers
- Poly(amides) (63)
- Poly(imides) (3)
- Poly(ureas) (4)
- Poly(ureas) (4)
- Poly(phosphazenes) (4)
- Poly(silanes) (1)
- Poly(silazanes) (1)

4.Main-chain Hetrocyclic Polymers
- Poly(acetals) (3)
- Poly(carboranes) (2)
- Poly(piperazines) (2)
- Poly(oxadiazoles) (2)

TABLE I: List of the 440 polymers in the polymer database.

TABLE II: Description of the fields in the database for each polymer.

No	Field Name	Data Type	Description
1	ID	Auto No	Primary Key
2	Polymer_Class	Text	Class of Polymer
3	Polymer_Name	Text	IUPAC Polymer Name
4	Mer_Chemical_Formula	Text	Chemical Formula of Mer
5	No_C_atoms	Number	No of C Atom Bonds in Mer
6	No_C-C_bonds	Number	No of C-C Single Bonds in Mer
7	No_C-H_bonds	Number	No of C-H Single Bonds in Mer
8	No_C-2-C_bonds	Number	No of C-C Double Bonds in Mer
9	No_C-3-C_bonds	Number	No of C-C Triple Bonds in Mer
10	No_H_atoms	Number	No of H Atom Bonds in Mer
11	No_Si_atoms	Number	No of Si Atom Bonds in Mer
12	No_Halide_atoms	Number	No of Halide Atom Bonds in Mer
13	No_P_atoms	Number	No of P Atom Bonds in Mer
14	No_N_atoms	Number	No of N Atom Bonds in Mer
15	No_O_atoms	Number	No of O Atom Bonds in Mer
16	No_S_atoms	Number	No of S Atom Bonds in Mer
17	No_TransitionMetal_atoms	Number	No of Transition Metal Atom Bonds in Mer
18	No_Cyclic_Rings	Number	No of Cyclic Rings in Mer
19	Mol_Wt_Num_Avg	Number	No average molecular weight
20	Mol_Wt_Wt_Avg	Number	Weight average mol weight
21	T_alpha	Number	T_α (^{o}K)
22	T_beta	Number	T_β (^{o}K)
23	T_gamma	Number	T_γ (^{o}K)
24	T_delta	Number	T_δ (^{o}K)
25	T_alpha_T_gamma	Number	T_α / T_γ
26	Mer_Aspect_Ratio	Number	Aspect Ratio of Mer
27	Mer_3d_Weiner_Number	Number	3-D Weiner Number of Mer
28	Dynamic_Modulus	Number	Dynamic Modulus (20^{o}C, dynes/cm^2)
29	T_m	Number	Melting Point (^{o}K)
30	Data_Status_Code	Number	Data Status Code
31	Blend_Status_Code	Number	Blend Status Code

parentheses gives the number of different individual polymers in each class. Thus the total data set consisted of 440 individual polymers. For each polymer, information stored included its molecular weight (polydispersity), mechanical and thermal properties, chemical structure and data reliability number. TABLE II gives the description and format of the fields. Fields 15 to 18, and 21 were average reported values from the best 4 sources. This is because these values depend on the technique used to find them and the polydispersity of the samples. The last two fields are codes (integers from 1 to 10) that reflect the quality of the data and the extent to which the different blend systems for the given polymer have been characterized, respectively. Numbers less than five suggest that the information is unreliable due to being old, or due to the fact that the experimental or theoretical techniques had a high source of error. All polymers had data status codes greater than five. The data status codes (DSC) were calculated using

$$DSC = \left\lfloor \frac{1}{4}\sum_{<1,4>}\frac{1}{CV_i}\times 10 \right\rfloor \tag{1}$$

where

$$CV_i = \frac{1}{4}\sum_{<1,4>}\frac{s_i}{y_i}\times 100\,. \tag{2}$$

s_i is the standard deviation, y_i and is the mean value for each of the fields 15 to 18, and 21 for each polymer. Thus CV_i is the average value of the coefficient of variance for each of the five fields 15 to 18 and 21. Note that for each polymer CV_i was never greater than 10%. The blend status code (BSC) of each polymer indicates the number of characterized blends having the polymer as one component and a DSC value of at least 5 for the given blend. Every polymer has a BSC of at least one. If a given polymer has more than 10 characterized blends with it as a component and with the DSC value of each of these at least 5 the BSC value is set to 10. Such a polymer obviously has its blends well characterized. The use of these codes ensures that the dataset used is reliable when selecting which polymer to include in the dataset and to ensure that all the polymers in the dataset meet a minimum criterion.

Fields 1 to 14 are self-explanatory. T_α is the primary glass-transition temperature at which the onset of long-range segmental mobility occurs[7] while T_β and T_γ with T_δ are lower order relaxation temperatures associated with motions in the back-bone and side-chain respectively[8]. The aspect ratio[9,10] is a 2-D measure of the asymmetry in a monomer (polymer repeat unit) and is the ratio of the length of the long axis to the short axis of the monomer. Monomers are three-dimensional objects and hence in addition to their topological and combinatorial content their 3-D character is of profound importance. The 3-D Weiner number is based on the 3-D (geometric, topographic) distance matrix, whose elements represent the shortest Cartesian distance between two $i-j$ pairs. The matrix is real and symmetric. The number is extracted from the matrix as indicated in reference [11]. For engineering plastics, the four transition temperatures together with the melting temperature, the aspect and Weiner numbers, and the elastic modulus with the given polydispersity (weight and number average molecular weight) serve to provide a comprehensive description of the material from the thermal, structural and mechanical stand point. Thus the question of reducing the inputs during the training and final selection of the networks was not considered as all the inputs were extremely important and could not be discarded if theoretical accuracy in the modeling was desired.

2.1.1 Polymer Properties Selected

While many properties are desired in an engineering plastic, probably the most important is its impact resistance[12]. One indicator of good impact resistance is the T_α / T_γ ratio[13] and the dynamic elastic modulus[14]. The higher this ratio the better the impact resistance. However high impact resistance with almost no elastic properties results in a brittle polymer that has no commercial use. Thus in the complete description of the overall mechanical properties of a polymer these properties have to be included, and they serve to accurately describe the polymers mechanical behavior over the entire useful temperature range. While reinforcing a polymer and creating cross-linked or composite materials gives extremely good materials, there is a sacrifice in the optical properties. Such materials cannot be easily molded and lack any elastic properties, greatly reducing their use and marketability[15]. In this work we concentrate on two engineering plastics and their various modifications. The first, bisphenol-A polycarbonate (PC)[16], is commercially available as Lexan/Makrolon/Calibre and is widely used in bullet-proof glass. It has a T_α / T_γ ratio of 2.5 and a dynamic elastic modulus at $20\,^oC$ of 5.02×10^9 dynes/cm^2. The second, poly(2,6-dimethyl-1,4-phenylene oxide) (PPO)[17], is widely used in the electrical industry as insulation in wiring and armatures, and in capacitors and dielectrics. Its T_α / T_γ ratio is 1.7 and dynamic elastic modulus at $20\,^oC$ is 6.21×10^9 dynes/cm^2.

From these two polymers 24 different modified PC's and 19 different modified PPO's were selected carefully to present to the final trained networks. The modifications were such so as to carefully ensure that a range of possibilities was considered. This included replacing the backbone and/or side chain carbon atoms (C) by silicon atoms (Si). As in the case of poly(silicone) it is well known that Si substitutions not only dramatically improve the dynamic mechanical response of the polymer, but also is a favorable species for polymerization. Other more important substitutions are also considered. The effects of amide-type group (-CO-R-NH) substitutions in the main chain and side chain are important. Poly(amides) like Nylon 6 and Nylon 66 contribute to a major class of commercial materials. They have good electrical, optical and mechanical properties. Another important class of polymers are poly(urethanes) (-NH-CO-NH-) based on the amide-type group. These polymers have extensive commercial applications and have high impact resistance, and excellent high temperature mechanical properties. They are also used as foams, and they can be readily cross-linked, to form densely packed, yet flexible rubbers. The amide-type group is highly polar, a distinct advantage. This causes favorable intermolecular interactions to dominate resulting in a better van-der-Waals type of weak interactive force between the polymer chains that directly lead to greater dimensional stability. Further, the amide-type groups break the linearity of the polymer chain, resulting in better "excluded volume" types of interactions. The other types of substitutions were based purely on steric factors. In side chains the effects of replacing methyl ($-CH_2$) groups with longer linear molecules like ethyl ($-CH_2-CH_3$) or even pentyl ($-CH_2-CH_2-CH_2-CH_2-CH_3$) will result in changes in excluded volume interactions without changing the polarity of the polymer. In many cases an increase in impact resistance is indicated. Besides the amide-type groups mentioned above substitutions involving oxygen and halogen atoms are important cases that are also considered. A wide array of polymers like poly(esters) and poly(vinyl halides) indicate the importance of these atoms in the main and pendant side chains. For each of the modifications presented to the trained networks their attractive and repulsive interaction parameters were calculated and checked to see that they were in the range for the given polymer class, thus effectively screening out abstruse structures. Details of this are given in [1]. From over 40 different modifications of PC and PPO presented to the final trained networks several candidates were found with improved properties. While the structures of all these polymers are not given those with improved properties are presented and discussed in more detail.

2.2 Neural Networks Used

The neural networks used here included the standard supervised backpropagation network with one, two and three hidden layers. The architecture for this type of network is similar to that shown in FIGURE 1(a) but without the jump connection from the input to the output layer. The number of hidden layers we used ranged from one to three. The other backpropagation networks used were the jump connected type, and the recurrent networks with feedback. Also used were networks with two hidden slabs each with its own activation function. Specialized supervised networks used included the probabilistic type, the general regression type, and the polynomial net. The only type of unsupervised network used was the Kohonen network. Further, eight different activation functions were used. These were the standard logistic function ($f(x) = 1/(1 + e^{-x})$ ...(f1)), the linear function ($f(x) = x$ ...(f2)), the hyperbolic tangent functions ($f(x) = \tanh(x)$ ...(f3) and $f(x) = \tanh(1.5x)$ ...(f4)), the sine function ($f(x) = \sin x$ ...(f5)), the symmetric logistic function ($f(x) = 2/(1 + e^{-x}) - 1$ ...(f6)), the Gaussian function ($f(x) = e^{-x^2}$ ...(f7)) and the Gaussian compliment function ($f(x) = 1 - e^{-x^2}$ ...(f8)). We shall refer to these functions as f1 to f8 respectively, henceforth.

2.2.1 Network Types

2.2.1.1 Standard Backpropagation Networks

The first type of network considered was the standard backpropagation network with one, two and three hidden layers. The advantage of more hidden layers is that different activation functions can be selected. While the most commonly used activation is logistic, in many cases depending on the distributions in the training inputs several other functions or combinations of functions are known to perform better.

The next type of network used was the jump connected network[18]. This is a backpropagation network in which every layer is connected or linked to every previous layer. As shown in FIGURE 1(a)-(c) there is a choice of one to three hidden layers. The disadvantage of using more than one hidden layer is an increase in training time, but in many cases better models can be obtained.

The other type of backpropagation network used is a recurrent network[19] (FIGURE1(d)-(f)). The three types of recurrent networks used are: (i) in which the input layer is fed back into the input layer itself, (ii) in which the hidden layer is fed back into the input layer, and (iii) in which the output layer is fed back into the input layer. They are most successful when there is a time series in the input data. In the first type, the long-term memory remembers the new input data and uses it when the next pattern is processed. In the second type the long-term memory remembers the hidden layer, which contains features detected in the raw data or previous patterns. This is the most powerful network. In the last type, long-term memory remembers outputs previously predicted. If there is no temporal structure in the data it may not work well. Thus a recurrent network may respond to the same input pattern differently at different times, depending on the patterns that have been presented as inputs at earlier times. These networks are trained the same way as the standard backpropagation networks except that patterns must always be presented in the same order. Random selection is not allowed. During the training it would be interesting to see the effect of randomly presenting the training data (as in the standard backpropagation network) versus grouping the data into the sets according to polymer classes and groups and presenting the ordered data sequentially to the network (as in recurrent networks). The feedback link contains a feedback factor that needs to be adjusted carefully. This factor (feedback 1) tells what proportion of the neuron values goes into long term memory itself, and (feedback 2) what proportion of the neuron values in the current pattern from either the input, hidden or output layer (depending on the network type) are fed into the long term memory. Both of these values must add up to one. If more emphasis is to be placed on historical patterns a higher proportion of neuron values are put on feedback 1. If more emphasis is to be placed on recent patterns then feedback 2 is set greater than 0.5.

The last backpropagation network used was one with multiple hidden layers (FIGURE 1(g)-(i)). The hidden layers act like feature detectors. Different activation functions applied to the hidden layer detect different features as a pattern processes through a network. A Gaussian function (f7) in one hidden slab detects features in the mid-range of the data while the Gaussian complement (f8) in another hidden slab detects features in the upper and lower extremes of the data. Combining the results leads to better prediction. When each slab has different activation functions it offers three ways of viewing data. The output layer receives two different views of the data's features as detected in the hidden slabs plus the original inputs. Data that have flat distributions are ideally suited for these networks.

2.2.1.2 Probabilistic Networks

In addition to the above backpropagation networks, four other networks not based on the backpropagation algorithm but equally powerful were used. The first of these was the probabilistic network[20] (PNN). PNN's are known for their ability to train quickly on sparse data sets. A PNN separates data into a specified number of output categories. They are three layer networks wherein the training patterns are presented to the input layer and the output layer has one neuron for each possible category. There must be as many neurons in the hidden layer as there are training patterns. The network produces activations in the output layer corresponding to the probability density function estimator for that category. The highest output represents the probable category.

Two main calibration techniques are used here: iterative and genetic adaptive. Iterative calibration training for a PNN proceeds in two parts. The first part trains the network with the data in the training set. The second part uses calibration to test a whole range of smoothing factors, trying to converge on one that works best for the network created in the first part. Training is faster than when using the genetic adaptive option. With genetic adaptive calibration a genetic algorithm is used to find appropriate individual smoothing factors for each input as well as an overall smoothing factor. (The input-smoothing factor is an adjustment used to modify the overall smoothing factor to provide a new value for each input.) Training takes longer than when using the iterative option. Training for PNN nets using the genetic adaptive option also proceeds in two parts. The first part trains the network with the data in the training set. The second part uses calibration to test a whole range of smoothing factors, trying to hone in on a combination that works best on the test set with the network created in the first part. The genetic algorithm is looking for a smoothing factor multiplier for each input, which in essence is an individual smoothing factor for each input. At the end of training, the individual smoothing factors may be used as a sensitivity analysis tool: the larger the factor for a given input, the more important that input is to the model at least as far as the test set is concerned. Inputs with low smoothing factors are candidates for removal for later trial, especially if the smoothing factor gets set to zero. (If it goes to zero, the network has removed the input anyway.) The genetic adaptive option is used when the input variables are of different types and some may have more of an impact on predicting the output than others. The genetic adaptive method will produce networks, which work much better on the test set but will take much longer to train. Genetic algorithms use fitness functions and fitness value measures to determine which of the individuals in the population survive and reproduce. Thus, survival of the fittest causes good solutions to evolve. The fitness for a PNN is the number of incorrect answers that we are obviously trying to minimize. Since PNN networks work by comparing patterns based upon their distance from each other, several types of distance metrics can be used. We use the Euclidean distance metric.

2.2.1.3 General Regression Neural Network

Another network we use that is able to train quickly on sparse data sets is the General Regression Neural Network[21] (GRNN). GRNN's are a type of supervised network that are able to produce continuous valued outputs rather than discrete categories like PNN's. We found that GRNN's respond much better than backpropagation to many types of problems, and are especially useful for continuous function approximations. GRNN's can have multidimensional input, and they will fit multidimensional surfaces through data. They are three layer networks where there must be one hidden neuron for each training pattern. There are no training parameters such as learning rate and momentum as in backpropagation, but there is a smoothing factor that is applied after the network is trained. GRNN's work by measuring how far a given sample pattern is from patterns in the training set in N dimensional space, where N is the number of inputs in the problem. When a new pattern is presented to the network that input pattern is compared in N dimensional space to all of the patterns in the training set to determine how far in distance it is from those patterns. The output that is predicted by the network is a proportional amount of all of the outputs in the training set. The proportion is based upon how far the new pattern is from the given patterns in the training set. For example, if a new pattern is in a cluster with other patterns in the training set, the outputs for the new pattern are going to be very close to the other patterns in the cluster around it. As with a PNN the calibration criteria for a GRNN is the same viz.: iterative and genetic adaptive. Here too the Euclidean distance metric is used.

2.2.1.4 Group Method of Data Handling Neural Network

Another type of network we use is the Polynomial net, also known as the Group Method of Data Handling (GMDH) Net[22]. This technique was invented by A. G. Ivakhnenko and enhanced by A. R. Baron. In fact, the GMDH network is not like regular feed forward networks and was not originally represented as a network. The GMDH network is implemented with polynomial terms in the links and a genetic component to decide how many layers are built. The result of training at the output layer can be represented as a polynomial function of all or some of the inputs. GMDH works by building successive layers with complex links (or connections) that are the individual terms of a polynomial. Using linear and non-linear regression creates these polynomial terms. The initial layer is simply the input layer. The first layer created is made by computing regressions of the input variables and then choosing the best ones. The second layer is created by computing regressions of the values in the first layer along with the input variables. (Note that we are essentially building polynomials of polynomials.) Again, the algorithm chooses only the best. These are called survivors. This process continues until the network stops getting better (according to a pre-specified selection criterion). The resulting network can be represented as a complex polynomial description of the model. You may view the formula, which contains the most significant input variables. In some respects, it is very much like using regression analysis, but it is far more powerful than regression analysis. GMDH can build very complex models while avoiding over fitting problems.

GMDH contains several evaluation methods, called selection criteria, to determine when it should stop training. One of these, called Regularity, is similar to Calibration in that the network uses the constructed architecture that works the best on the test set. The other selection criteria do not need a test set because the network automatically penalizes models that become too complex in order to prevent over training. The advantage of this is that you can use all available data to train the network. A by-product of GMDH is that it recognizes the most significant variables as it trains, and one can get a list of them.

2.2.1.5 Kohonen Network

The last network used is of the unsupervised type and is the Kohonen network[23]. The Kohonen Self Organizing Map network used has the ability to learn without being shown correct outputs in sample patterns. These networks are able to separate data into a specified number of categories. There are only two layers: an input layer and an output layer which has one neuron for each possible output category. The training patterns are presented to the input layer, then propagated to the output layer and evaluated. One output neuron is the “winner”. The network weights are adjusted during training. This process is repeated for all patterns for a number of epochs chosen in advance. This network is very sensitive to learning rate. It is lowered slightly but steadily as the training progresses, causing smaller and smaller weight changes. This causes the network to stabilize. The network adjusts the weights for the neurons in a neighborhood around the winning neuron. The neighborhood size is variable, starting off fairly large (close to the number of categories) and decreasing with learning until during the last training events the neighborhood is zero, meaning by then only the winning neuron's weights are changed. By that time the learning rate is very small, and the clusters have been defined. The learning rate and neighborhood size are automatically adjusted, but the initial values as well as the total number of epochs that learning will continue for need to be specified. How well a Kohonen network classifies data is dependent upon how well these parameters are set. Learning may not be successful if there isn't a large enough neighborhood or if the training doesn't run for enough epochs, or if the learning rate is too small at the start. The pattern selection is rotation where each pattern is applied to the network one at a time.

2.2.2 Training Method

The most important criterion for network training and optimization is to ensure generalization. Extreme care must be taken to prevent the memorization of input data. Hence there must be an evaluation set created either from the training data or a separate set of data with known outputs. This must then be used to check the accuracy of the trained network. The process of selecting the final network, and the training and optimization involved however will involve a number of steps. For various values of momentum, learning rate, initial weight distribution, number of nodes in the various layers and the number of layers itself, a significant measure of trial and error is involved to test every type of network for the best combination of parameters. The training set must be varied enough and contain all the patterns so that the entire possible spectrum for the given problem can be represented. To stop the training the following are monitored and the training is stopped when they are within predefined values. The first of the two most important measures is the correlation coefficient "r" which is a statistical measure of the strength of the relationship between the actual versus the predicted results. The r coefficient ranges from -1 to +1. The closer r is to 1, the stronger the positive linear relationship, and the closer r is to -1, the stronger the negative linear relationship. When r is near 0 there is no linear relationship.

$r = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$, where $\sigma_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$, $\sigma_{xx} = \sum x^2 - \frac{(\sum x)^2}{n}$, and $\sigma_{yy} = \sum y^2 - \frac{(\sum y)^2}{n}$. N equals number of patterns, *x* refers to the set of actual outputs, and *y* refers to the predicted outputs.

The second coefficient R^2 is a statistical indicator usually applied to multiple regression analysis. It compares the accuracy of the model to the accuracy of a trivial benchmark model wherein the prediction is just the mean of all of the samples. A perfect fit would result in an R^2 value of 1, a very good fit near 1, and a very poor fit less than 0. If the neural model predictions are worse than predictions obtained by just using the mean of the sample case outputs, the R^2 value will be less than 0. $R^2 = 1 - \frac{\sigma_e}{\sigma_m}$, where $\sigma_e = \sum (y - \tilde{y}\}^2$ and $\sigma_m = \sum (y - \bar{y}\}^2$. $\tilde{y}$ is the predicted value of the actual value y, $\bar{y}$ and is the mean of the *y* values. Other parameters monitored but less important are percent of the network answers that are within 5%, 10%, 20%, 30%, and over 30% of the actual answers used to train the networks. The mean squared error, mean absolute error, minimum absolute error, and maximum absolute error are also tracked. For each type of Neural Network there is much experimentation needed until the best network is found and then among the different network types that can be used we have to select the one that is best for the given problem.

The inputs for training the backpropagation nets and the GRNN network taken from the polymer database shown in TABLE II were fields numbered 6 to 9, 11 to 18, 20, 26, and 27. The number of neurons in the input layer was the number of inputs in every case. The outputs for training of these types of nets were fields numbered 25 and 28, and were the number of neurons in the output layer. In all, the original pattern file had 440 patterns. 80% were extracted at random to form the training set, which now had 352 patterns. 20% were then in the testing set (88 patterns). It was found that in this case the multi-layer Backpropagation networks gave the best results as compared to the other type of networks mentioned above (PNN, GRNN, GMDH and Kohonen). In the case of the PNN and Kohonen networks the inputs were the same as above, however there was no need for outputs. After training and testing the inputs for the 24 modified PC's and 19 modified PPO's were presented to the final trained PNN network. The network classified each of these into one of the 440 polymers. Thus PNN and Kohonen networks provided only a rough estimate of the mechanical properties of the modified polymers. However this was a tremendous help to verify the viability of a particular modified polymer before being presented to the other networks. The modified PC's and PPO's were finally selected only if these nets successfully classed them in their polymer groups. The number of inputs and the number of neurons in the input layer of the GMDH network was the same as in the case of PNN and GRNN nets. However the number of outputs was always one. The first output was the field numbered 25 followed by 28. Thus two final GMDH networks were created to which the modified polymers were presented.

3. RESULTS AND DISCUSSIONS

As shown in FIGURE 1 and discussed above there are several networks to use. In the end one network with a given set of parameters must be selected as the best to use to make the predictions. This involves a considerable amount of trial and error and training with all the different networks for all possible values of the parameters. Another method would be to take the predictions from the various best nets found for each type of network and average the values. However in this case there is a drawback in that the averaging process can actually cause the deterioration of the final predicted values. As discussed in detail in our earlier paper[1] polymer blend miscibility is very sensitive to the interaction parameters. These must be predicted extremely accurately for predicting miscibility. In this case we have seen that the ratio T_α / T_γ is extremely sensitive and important to predicting good mechanical properties. Thus we do not average our predictions of this value with predictions from the best nets we find for the different network architecture we use but rather select one network from a single network architecture that we use for our final predictions. For the sake of comparison we also present results from the other network architectures. The main advantage of this comparison enables us to select networks that provide quick training with fairly acceptable results as compared to the best network we find. Further, some network architectures like the Kohonen and PNN networks serve a useful purpose in deciding which modified polymers are useful to investigate further.

3.1 Final Networks

The best network that we found for this problem was the multi-layer backpropagation network which gave the best results. The bottom-up technique for selecting the number of hidden nodes was used in favor of the top-down method. The number of calibration events was 50 and training was saved based on the best test set. The training ended when the number of events since the minimum average error exceeded 500,000. Weight updates selection was momentum. The weight updates not only include the change dictated by learning rate, but include a portion of the last weight change as well. Pattern Selection was random during training.

FIGURE 2 shows the effects of varying the number of hidden nodes for a single hidden layer backpropagation network. From this we see that r^2, the mean squared error and the correlation coefficient all were best for 17 hidden nodes and a very high number of hidden nodes did not appear to improve these values appreciably further. Thus the number of hidden nodes was fixed at 17. In this region the ‘Percent above 30%’ (of the actual values) also showed a minimum while the correlation coefficient exhibited a maximum. In FIGURE 3 the results of varying the initial weights are given, while in FIGURE 4 the effects of varying momentum are shown. FIGURE 5 shows the effect of varying learning rate. The activation functions f1, f3, f4 and f6 were tried. f1 gave the best results. As will be shown,

TABLE III: Effect of adding a second layer to the standard backpropagation single hidden layer neural network. All activation functions are logistic (f1). Input scaling function is logistic (f1). Number

of hidden nodes in first hidden layer = 17. Learning Rate = 0.05. Momentum = 0.5. Initial Weights = 0.3.

Second Hidden Layer Nodes	5	10	15
R^2	0.854	0.919	0.893
r^2	0.913	0.922	0.919
Mean Squared Error	0.072	0.068	0.069
Mean Absolute Error	0.285	0.226	0.225
Min. Absolute Error	0.008	0.007	0.007
Max. Absolute Error	0.300	0.285	0.289
Correlation coefficient	0.924	0.936	0.939
Percent within 5%	32.56	34.58	34.74
Percent within 5% to 10%	32.27	33.04	32.95
Percent within 10% to 20%	17.54	18.59	19.01
Percent within 20% to 30%	8.32	7.38	6.39
Percent over 30%	9.31	6.41	6.91
Training Time (epochs)	3,478,000	3,965,000	4,219,000

the addition of a second hidden layer did not improve the results. For the ‘Percent within 5%’ and the ‘Percent over 30%’ values the case when the number of nodes was equal to 17 (N_{17}) gave better results than the case when the number of nodes was equal to 100 (N_{100}). However the N_{100} case gave marginally better R^2, r^2, and correlation coefficient values. The N_{17} case also had the lowest mean squared error and mean absolute error. Given such narrow differences made making choices difficult, but finally considering the fact that higher nodes may cause redundancy in many weights, and that the N_{17} case did have the lowest mean square error, mean absolute error and better ‘Percent within 5%’ and ‘Percent over 30%’ values as compared to the N_{100} case, it was selected. It also had the second highest value for the correlation coefficient. Also the graphs in FIGURE 2 tended to flatten out after 15 nodes, suggesting some statistical fluctuations for nodes greater than 15. This indicates that the (near) optimum number of nodes for accurately defining the given problem was reached. Genetic algorithms or other combinatorial optimization methods could also be effectively used, but in terms of simplicity this was the best approach. FIGURE 3 shows the results obtained when varying the initial weights from 0.3. The number of hidden nodes was now fixed at 17. All other parameters including the learning rate and momentum were the same as in the case above. As compared to the case of 0.3 for 17 nodes in FIGURE 2 all the values for correlation coefficient, r^2, and mean squared error are poorer. FIGURE 4 gives the results of varying the momentum with learning rate fixed. All other parameters were the same as in the first case with the number of hidden nodes fixed at 17. Momentum of 0.4 gave only a marginally poorer value of correlation coefficient, r^2, and mean squared error as compared to a momentum of 0.5. Lower or higher values of momentum gave poor results. FIGURE 5 gives the results of varying learning rate keeping momentum fixed at 0.5. The number of hidden nodes was 17, and all other parameters were the same as in the first case. Increasing the learning rate caused further deterioration in the values of correlation coefficient, r^2, and mean squared error, from gradual to more rapid as the learning rate increased.

TABLE IV: Details of final selected model.

Type of Neural Network	Standard Backpropagation
Number of Hidden Layers	1
Number of input nodes	14
Number of hidden nodes	17
Number of output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.3
Learning Rate	0.05
Momentum	0.5
Percent over 30%	9.31
Training Time	1,573,000 epochs

Finally adding a second hidden layer to the network did not show any improvements as can be seen from TABLE.III. All parameters are the same as in the first case. The number of hidden nodes in the first hidden layer is 17. While increasing the number of hidden nodes in the second layer did improve the results they were in all cases poorer than the final selected model. FIGURE 3(a) and FIGURE 3(d) show the reason an initial weight of 0.3 was chosen in the final model. This value also gave the smallest mean squared error (FIGURE 3(c)), and highest 'Percent < 5%' values (FIGURE 3(b)). For momentum of 0.5 FIGURE 4(a) gave a maxima, and the mean squared error was minimum (FIGURE 4(c)). For this value as seen in FIGURE 4(d) the graph flattened out. From FIGURE 4(b), 'Percent < 5%' gave a maximum with a corresponding minimum for 'Percent > 30%'. Thus this value appeared to be an optimal number and was selected in the final model. FIGURE 5 shows that a choice of learning rate of 0.05 leads to the deterioration of the results with an increase in learning rate. The details of the final selected model are given in TABLE.IV. The min/max values of each of the input variables were set by scanning the input before being applied to the network. File extraction was random. FIGURE 6 gives a scatter plot of the actual versus the predicted output when the trained network is applied to the production file with 88 patterns randomly selected from the 440 patterns in the database where the exact values are known and can be compared with the predicted values from the final model. Note that this set differed from the 88 pattern testing set. In FIGURE 6 points on the straight line indicate that the actual and predicted output were identical. If most of the points lie on this line, there is a danger of memorization and lack of generalization by the network. Similarly wide deviations from the straight line indicate a poorly trained network that may not give accurate predictions.

TABLE.V-TABLE.VIII give the details of the best nets for the other network architectures. While the final predictions using these were not as good as the case when the final selected model was used (TABLE.IV) several important points were noted. In case of the jump connection nets (TABLE.V) the training time was higher the more the number of hidden layers. However the number of nodes in each hidden layer also increased. This indicated that jump connections added a degree of complexity that was not particularly useful in this case. In case of the

TABLE V: Details of best nets for the different neural network architectures. (a) Three layers with a jump connection. (b) Four layers with jump connections. (c) Five layers with jump connections.

a)	
Type	Three layers - jump connection
Hidden Layers	1
Input nodes	14
Hidden nodes	86
Output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.1
Learning Rate	0.025
Momentum	0.2
Training Time	4,678,000 epochs
b)	
Type	Four Layers - jump connection
Hidden Layers	2
Input nodes	14
First hidden layer nodes	86
Second hidden layer nodes	138
Output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.15
Learning Rate	0.01
Momentum	0.3
Training Time	5,346,000 epochs
c)	
Type	Four Layers - jump connection
Hidden Layers	2
Input nodes	14
First hidden layer nodes	86
Second hidden layer nodes	138
Third hidden layer nodes	215
Output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.2
Learning Rate	0.005
Momentum	0.4
Training Time	12,387,000epochs

recurrent networks (TABLE.VI) the final network shown in TABLE.VI(c) was the network that trained

TABLE VI: Details of best nets for the different neural network architectures. (a) Recurrent Net with input layer dampened feedback. (b) Recurrent Net with hidden layer dampened feedback. (c) Recurrent Net with output layer dampened feedback.

a)	
Type	Recurrent Net (Input layer feedback to input layer)
Hidden Layers	2
Input nodes	14
First hidden layer nodes	40
Second hidden layer nodes	40
Output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Feedback factor	0.5
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.1
Learning Rate	0.05
Momentum	0.2
Training Time	967,000 epochs
b)	
Type	Recurrent Net (Hidden layer feedback to input layer)
Hidden Layers	2
Input nodes	14
First hidden layer nodes	32
Second hidden layer nodes	32
Output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.2
Learning Rate	0.05
Momentum	0.3
Training Time	1,873,000 epochs
c)	
Type	Recurrent Net (Output layer feedback to input layer)
Hidden Layers	2
Input nodes	14
First hidden layer nodes	30
Second hidden layer nodes	30
Output nodes	2
Activation function	Logistic (f1)
Scaling function	Logistic (f1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.3
Learning Rate	0.05
Momentum	0.4
Training Time	902,000 epochs

the fastest as compared to the rest of the nets used. The predictions using this network however were not better than those for the final selected model (TABLE.IV). TABLE.VII gives the details of the final nets with two hidden layers with the possibility of using different activation functions in the two hidden layers. A typical choice of activation function would be f7 or f8. However the final model in this case did not find any hidden features or improve the final predictions. In TABLE.VIII the best GRNN and GMDH networks found are listed. However in this case both the nets did not give better results than the final best model. Also training times were as high as in the case of jump connected nets. Note that all training times given for the various best nets are rounded to the nearest 1000 epochs. An epoch is one complete training cycle for a given set of inputs until the next set of weight adjustments (backpropagation nets) or in the case of GRNN and GMDH nets coefficient adjustments. To compare the different nets and select the final model the 88 pattern testing set was used. Training time was not a concern here as accuracy in prediction was more a concern rather than speed of training. Clearly the best network was the final model which gave slightly better values consistently for all the statistical parameters considered as compared to the other best nets (FIGURE 7). Thus although for one or two statistical parameters other models were slightly better, the selected final model clearly was consistently better, overall.

3.2 Predicted Properties

TABLE.IX gives the results when the final selected model is used. FIGURE 8 gives their structures. We find that for these modifications better T_α / T_γ ratios and dynamic mechanical modulus values were predicted as compared to the parent polymers. For the other cases both the values were poorer than that of the parent. On the analysis of these results we were able to conclude that both steric factors, and the intra- and inter-

TABLE VII: Best nets. (a) Two hidden slabs with two activation functions. (b)Three hidden slabs with three activation functions. (c)Two hidden slabs with two activation functions and a jump connection.

a)	
Type	Two hidden layers
Hidden Layers	2
Input nodes	14
First hidden layer nodes	35
Second hidden layer nodes	35
Output nodes	2
Activation function (first hidden layer)	f7
Activation function (second hidden layer)	f8
Scaling function	Logistic
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.5
Learning Rate	0.01
Momentum	0.1
Training Time	5,207,000 epochs
b)	
Type	Three hidden layers
Hidden Layers	3
Input nodes	14
First hidden layer nodes	55
Second hidden layer nodes	225
Third hidden layer nodes	55
Output nodes	2
Activation function (first hidden layer)	f7
Activation function (second hidden layer)	f1
Activation function (third hidden layer)	f8
Scaling function	Logistic
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.5
Learning Rate	0.01
Momentum	0.1
Training Time	5,993,000 epochs
c)	
Type	Two hidden layers with a jump connection
Hidden Layers	2
Input nodes	14
First hidden layer nodes	90
Second hidden layer nodes	90
Output nodes	2
Activation function (first hidden layer)	f7
Activation function (second hidden layer)	f8
Scaling function	Logistic
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.5
Learning Rate	0.05
Momentum	0.1
Training Time	6,007,000 epochs

TABLE VIII: Details of best nets for the different neural network architechtures. (a) GRNN. (b) GMDH.

a)	
Type	GRNN
Hidden Layers	1
Input nodes	14
Hidden layer nodes	440
Output nodes	2
Calibration	Iterative
Training Time	24,003,000 epochs
b)	
Type	GMDH
Hidden Layers	1
Input nodes	14
Output nodes	1
Selection Criteria	Regularity[22]
Training Time	35,325,000 epochs

TABLE IX: a) Results of applying the final model to 24 modified bisphenol-A polycarbonates. b) Results of applying the final model to 19 modified poly(2,6-dimethyl-1,4-phenylene oxide)

Monomer	T_α / T_γ	Dynamic Modulus (20^{o}C, dynes/cm^2)
a)		
Modification PC-1	3.13	5.67×10^9
Modification PC-2	2.70	5.22×10^9
Modification PC-3	2.74	5.38×10^9
Modification PC-4	3.37	6.39×10^9
Modification PC-5	2.58	5.06×10^9
b)		
Modification PPO-1	1.98	6.32×10^9
Modification PPO-2	2.29	6.59×10^9

molecular polarities of the polymer play a vital role in the final outcome of the prediction of the mechanical properties of the polymers tested. From steric considerations we were definitely able to conclude that symmetry and slight increases in the excluded volume improved the mechanical properties. Thus modification PC-5 with symmetric butyl groups ($-CH_2-CH_2-CH_2-CH_3$) gave an improvement, but when these groups were pentyl or higher there was a drastic loss in the dynamic mechanical modulus. Similarly removing one such group had a detrimental effect on the mechanical properties. We also predicted better mechanical properties when Si substitutions were made, as can be seen in the case of modification PC-2, modification PC-3, and modification PPO-1. Silicon like carbon exhibits sp^3 hybridization. Both have similar ionization potentials. However silicon has a bigger atomic radius and hence larger bonding and anti-bonding orbitals, resulting in larger bond lengths. This appears to increase the excluded volume interactions, which directly enhances the weak van-der-Waals type attractive forces between the polymers. As a result we see a better mechanical response. However there appears to be a critical limit beyond which increases in the excluded volume actually begin to have a reverse effect. The addition of oxygen, or halide atoms on main or pendant side groups did not result in better mechanical properties. The strong dependence of good mechanical properties due to Si substitutions is established from the results of modification PC-1, 2, 3, and 5 and modification PPO-1. Also we have found that substitutions involving amino-type groups with the constraint that they are symmetric on the backbone (modification PC-4, and modification PPO-2) favor an increase in the mechanical properties as compared to the parents. In fact the polarities and the resulting charge distributions of the mer appear to be more important than steric factors. This again reiterates the fact that polymer dynamics in the bulk is mainly characterized by inter-molecular interactions like the weak van-der-Waals type of attractive forces. Thus steric factors though important appear to be secondary factors in polar polymers like PC and PPO. This is one reason why we now use these modifications[1], where blend miscibility is governed by intra- and inter-molecular interactions in the bulk between the polymer species through the models on blending. However the effects of steric factors were also evaluated[1].

4. CONCLUSIONS

We show how neural networks can be successfully used to predict polymer properties. We have also demonstrated that the complex structure-property relationships in polymers can be captured effectively by neural networks. In addition we can use nets to fine tune polymers to exhibit desired properties. While Kohonen and PNN nets provide a general indication of feasibility of a given polymer modification, extremely specialized nets can be built both with a high degree of accuracy or quick training to predict polymer properties. Further the large amount of information available on polymers can be used. While not presented here, the study of the detailed complex polynomials and the coefficients built by GRNN and GMDH nets help to quantify the structure-property relationships between the different classes of polymers. These can easily be compared to theoretical models built like the Takayanagi model for the mechanical properties of polymers[9]. Finally as we have shown[1], we are not restricted to using only this method exclusively but can complement other techniques in very important ways, and neither are we restricted to a particular set of parameters, but can study other areas of polymer design including polymerization kinetics, stability and degradation to name just a few.

REFERENCES

[1] N. K. Roy, D. P. Landau, and W. D. Potter, Applied Intelligence **20(3),** 215 (2004).

[2] D. Muller and J. Reinhardt, *Neural Networks: An Introduction* (Springer-Verlag, New York, 1990).

[3] S. Haykin, *Neural Networks: A Comprehensive Foundation* (Macmillan College Publishing Company, New York, 1994).

[4] D. W. Heerman, *Computer Simulation Methods in Theoretical Physics* (Springer-Verlag, Heidelberg, 1986).

[5] M. V. Volkenstein, *Configurational Statistics of Polymer Chains* (Wiley (Interscience), New York, 1963).

[6] G. Odian, *Principles of Polymerization* (McGraw-Hill, New York, 1970).

[7] L. H. Peebles, *Molecular Weight Distribution in Polymers* (Interscience, New York, 1971).

[8] R. M. Ogorkiewicz, *Engineering Properties of Thermoplastics* (John Wiley and Sons, New York, 1970).

[9] D. W. van Krevelen and P. J. Hoftyzer, *Properties of Polymers: Their Estimation and Correlation with Chemical Structure* (Elsevier Science Publishers, Amsterdam, 1976).

[10] J. M. Schultz, *Polymer Materials Science* (Prentice Hall, Englewood Cliffs, New Jersey, 1974).

[11] N. Trinajstic, *Chemical Graph Theory* (CRC Press, London, 1992).

[12] I. M. Ward, *Mechanical Properties of Solid Polymers* (Wiley-Interscience, New York, 1989).

[13] I. E. Neilsen, *Mechanical Properties of Polymers* (Reinhold, New York, 1991).

[14] T. Murayama, *Dynamic Mechanical Analysis of Polymeric Materials* (Elsevier, New York, 1978).

[15] E. E. Bear, *Engineering Design for Plastics* (Reinhold, New York, 1964).

[16] H. Schnell, *Chemistry and Physics of Polycarbonates* (Wiley, New York, 1964).

[17] J. J. Heijbouer, Polym.Sci.Polym.Symp., C **16**, 3755 (1968).

[18] P. Wasserman, *Neural Computing, Theory and Practice* (Van Nostrand Reinhold, New York, 1989).

[19] Y. Nishikawa, H. Kita, and A. Kawamura, Proceedings of the IEEE International Conference on Neural Networks **1**, I 684 (1990).

[20] D. Specht, Proceedings of the IEEE International Conference on Neural Networks **1**, 525 (1988).

[21] D. Specht, IEEE Trans. on Neural Networks **2**, 6 568 (1991).

[22] E. S. J. Farlow, Statistics:Textbooks and Monographs **54** (1984).

[23] M. Caudill, Neural Network Primer, AI Expert **25** (1990).

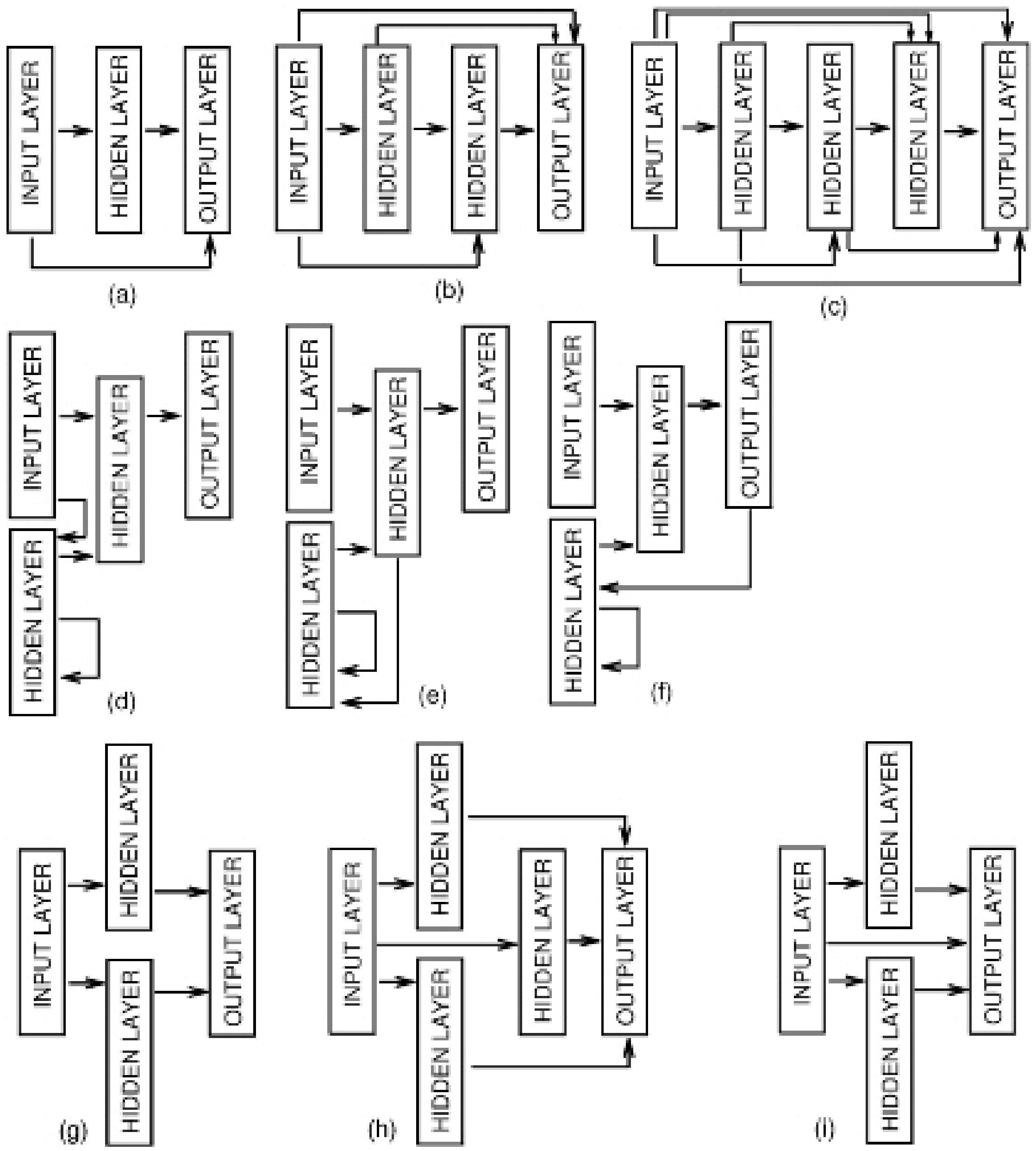

FIGURE 1: Neural Network architechtures based on the standard backpropagation algorithm: (a) Three layers with a jump connection. (b) Four layers with jump connections. (c) Five layers with jump connections. (d) Recurrent Net with input layer dampened feedback. (e) Recurrent Net with hidden layer dampened feedback. (f) Recurrent Net with output layer dampened feedback. (g) Two hidden slabs with possibility of two activation functions. (h) Three hidden slabs with possibility of three activation functions. (i) Two hidden slabs with possibility of two activation functions and a jump connection.

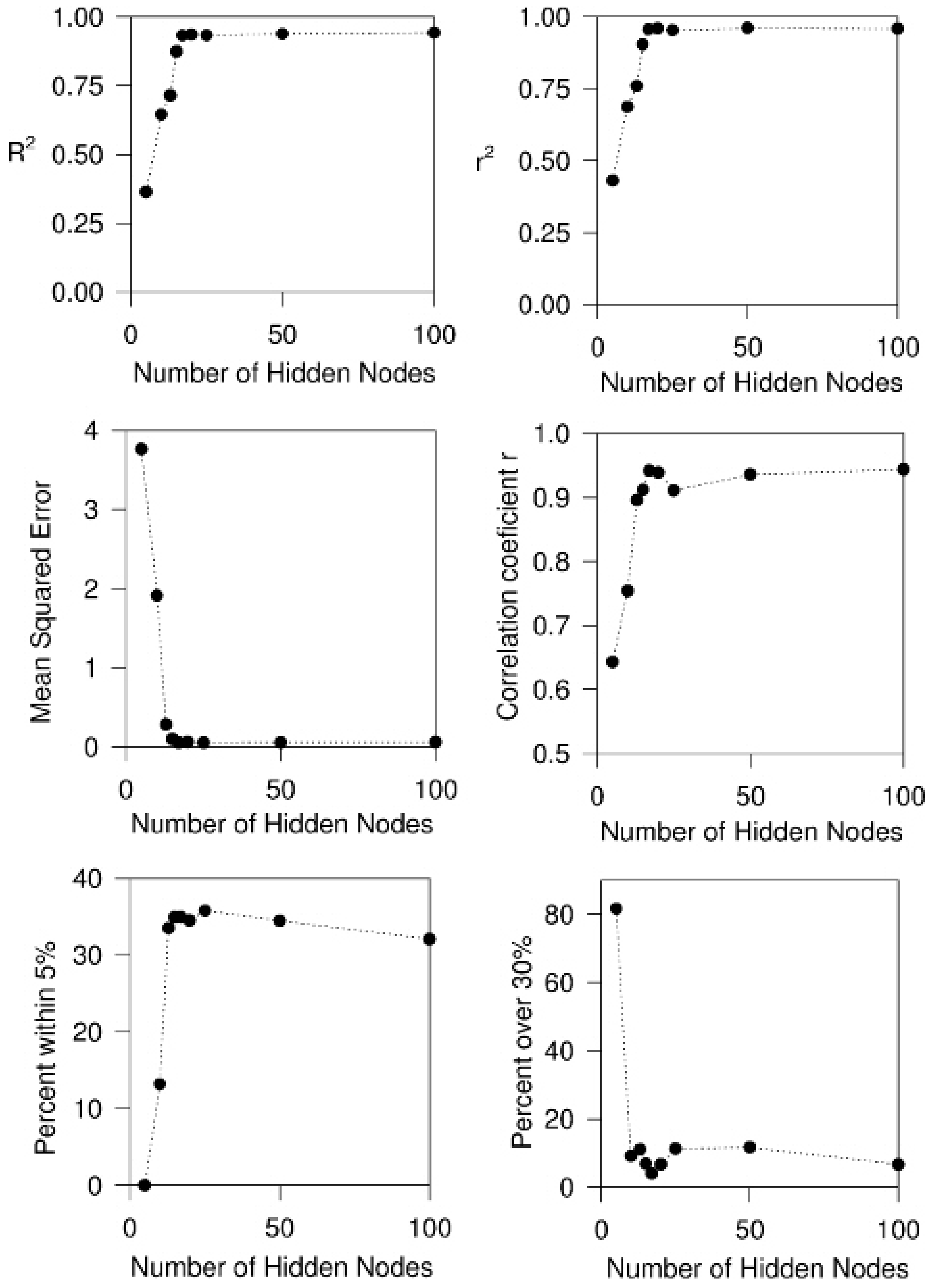

FIGURE 2: Error plots for variation of number of hidden nodes in a single hidden layer backpropagation neural network.

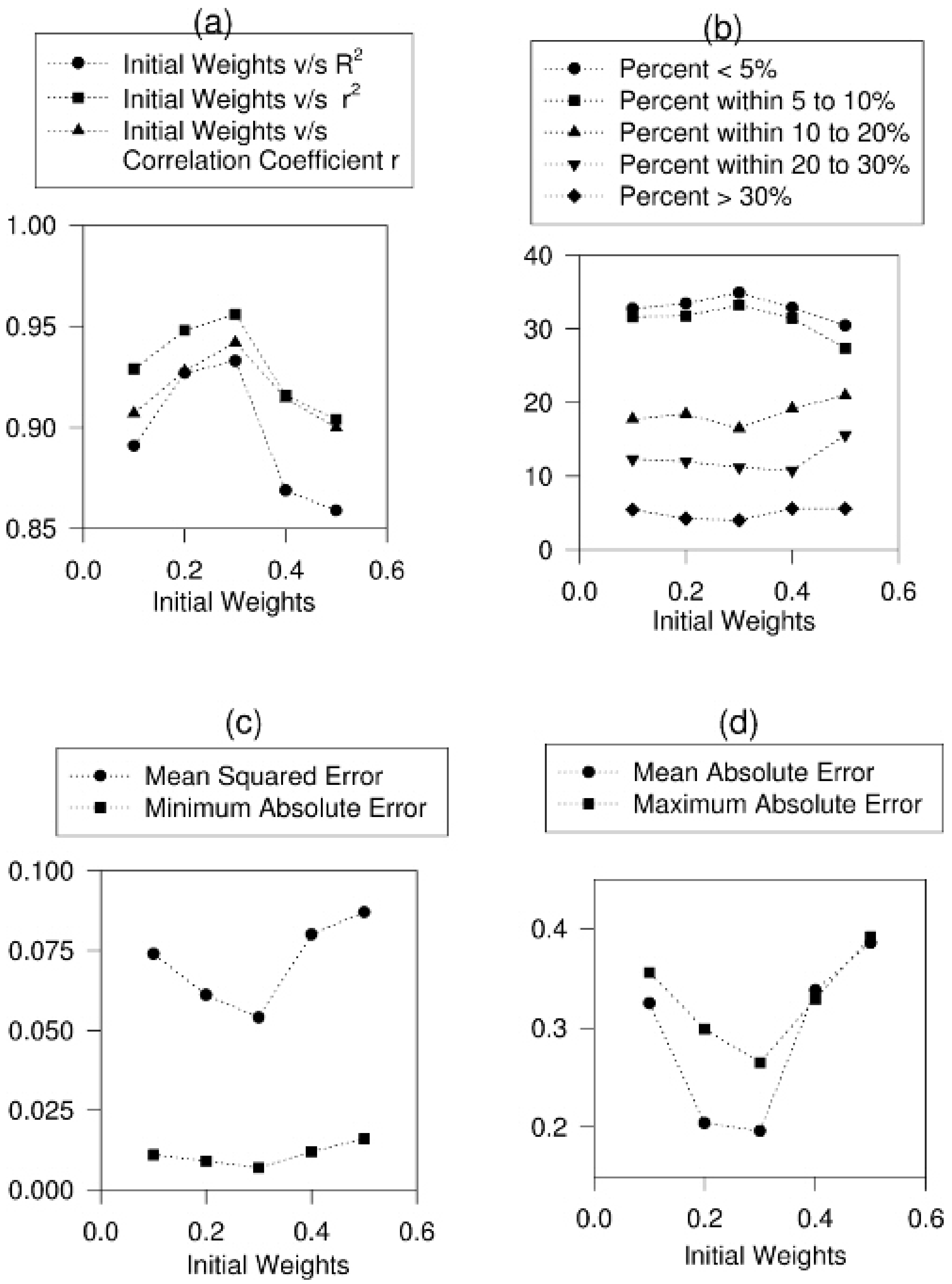

FIGURE 3: Effect of varying the initial weights.

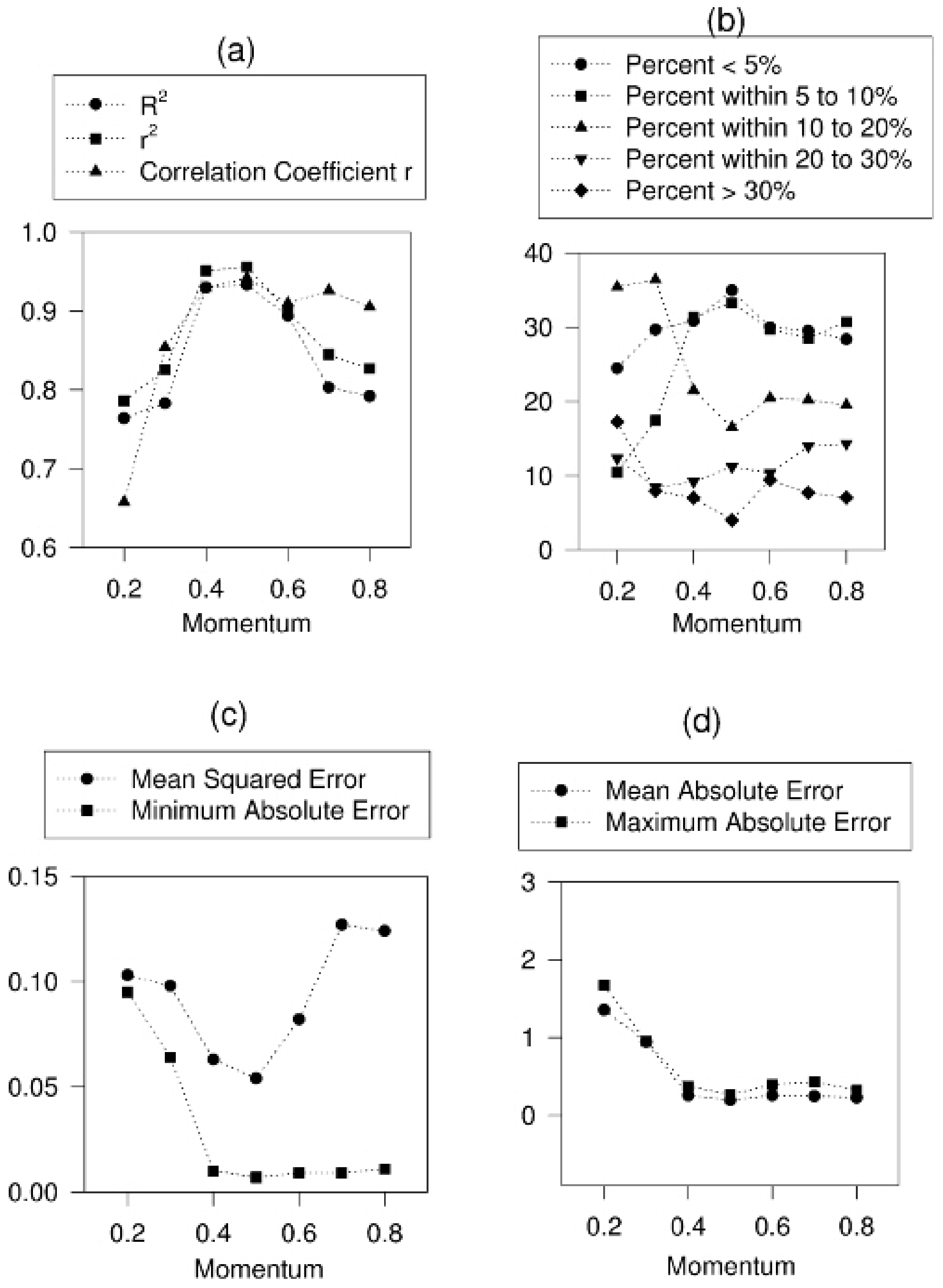

FIGURE 4: Effect of varying momentum.

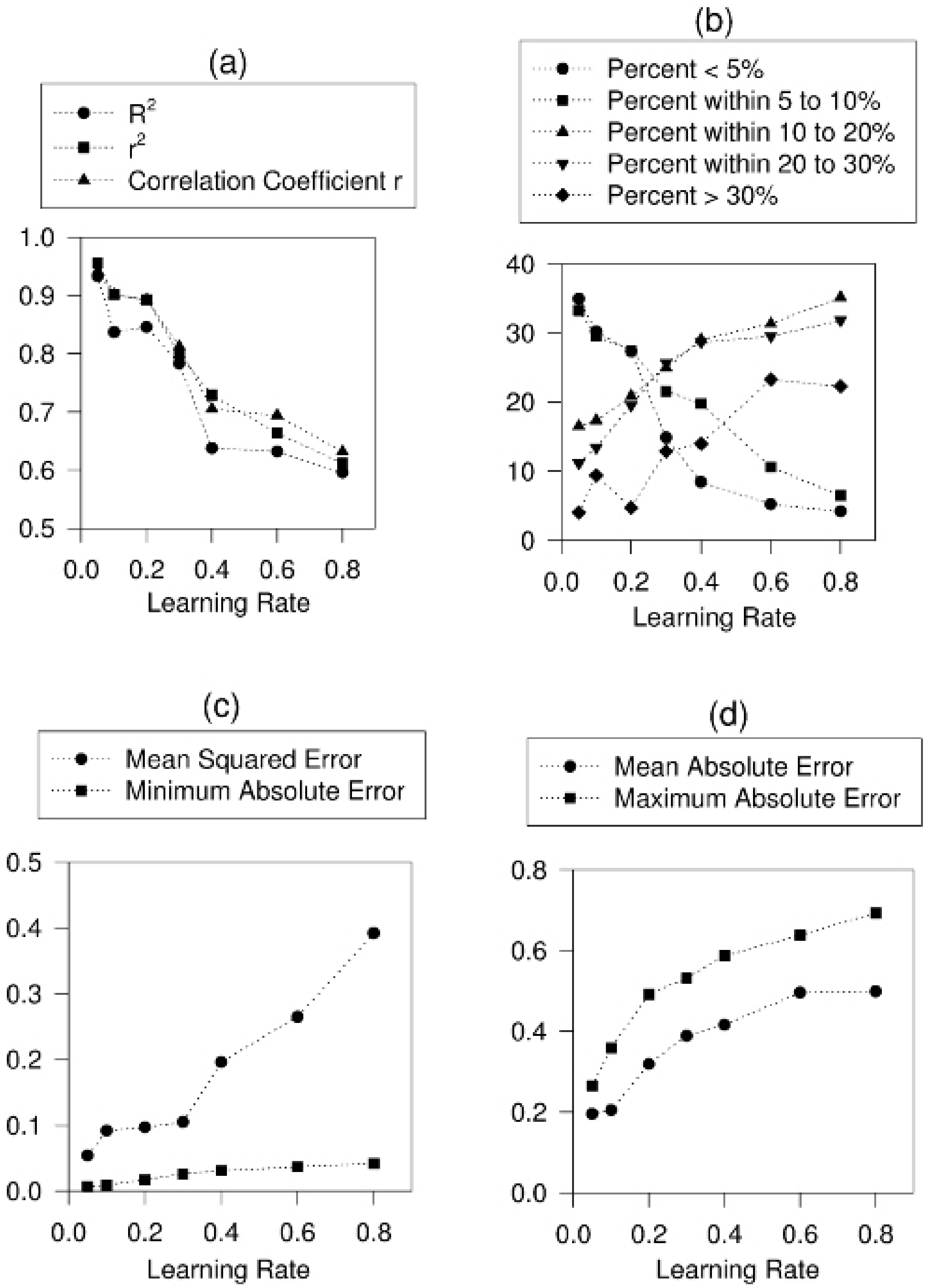

FIGURE 5: Effect of varying learning rate.

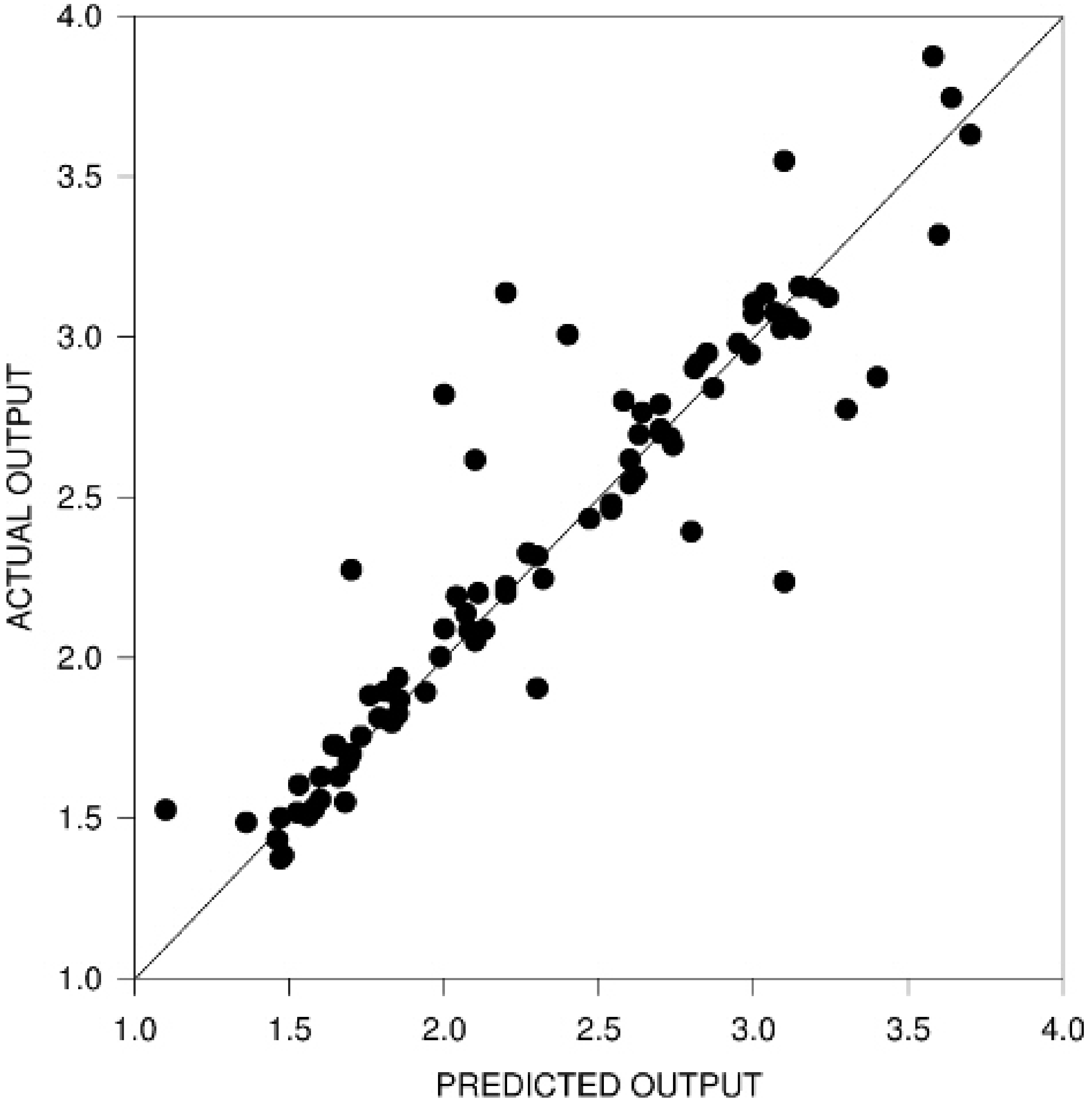

FIGURE 6: Actual versus predicted output when final model is applied to the 88 pattern testing set.

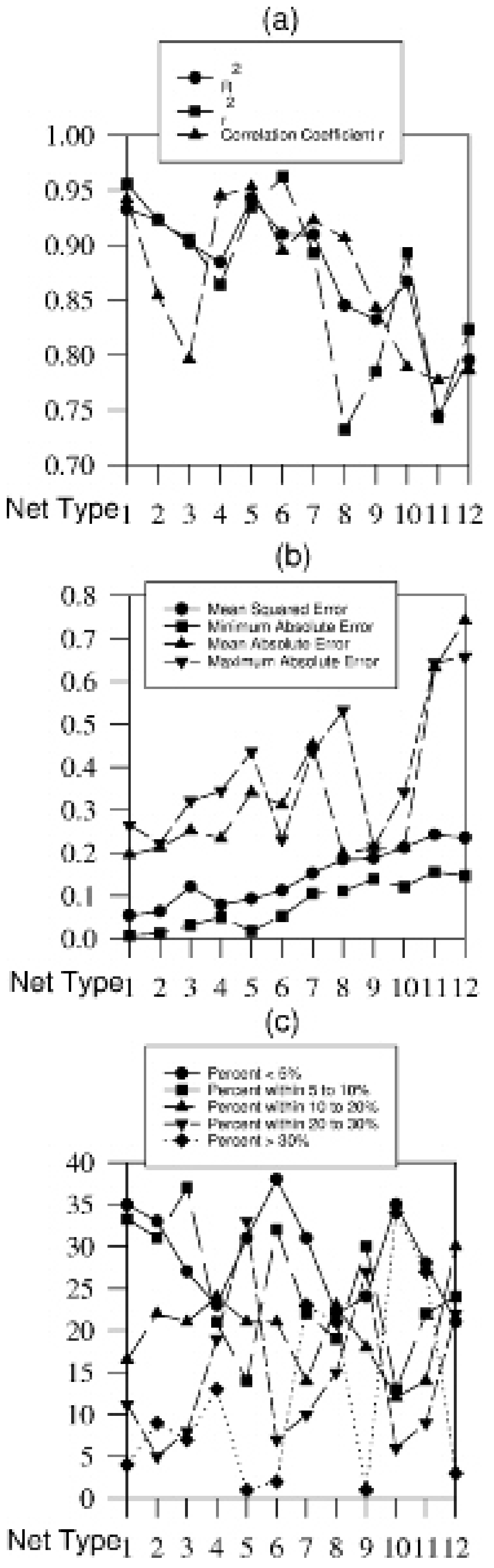

FIGURE 7: Comparisons of the different best nets. (Net Type: 1.Final Selected Model 2.Three layers with a jump connection. 3.Four layers with jump connections. 4.Five layers with jump connections. 5.Recurrent Net with input layer dampened feedback. 6.Recurrent Net

with hidden layer dampened feedback. 7.Recurrent Net with output layer dampened feedback. 8.Two hidden slabs with possibility of two activation functions. 9.Three hidden slabs with possibility of three activation functions. 10.Two hidden slabs with possibility of two activation functions and a jump connection. 11.GRNN. 12.GMDH.)

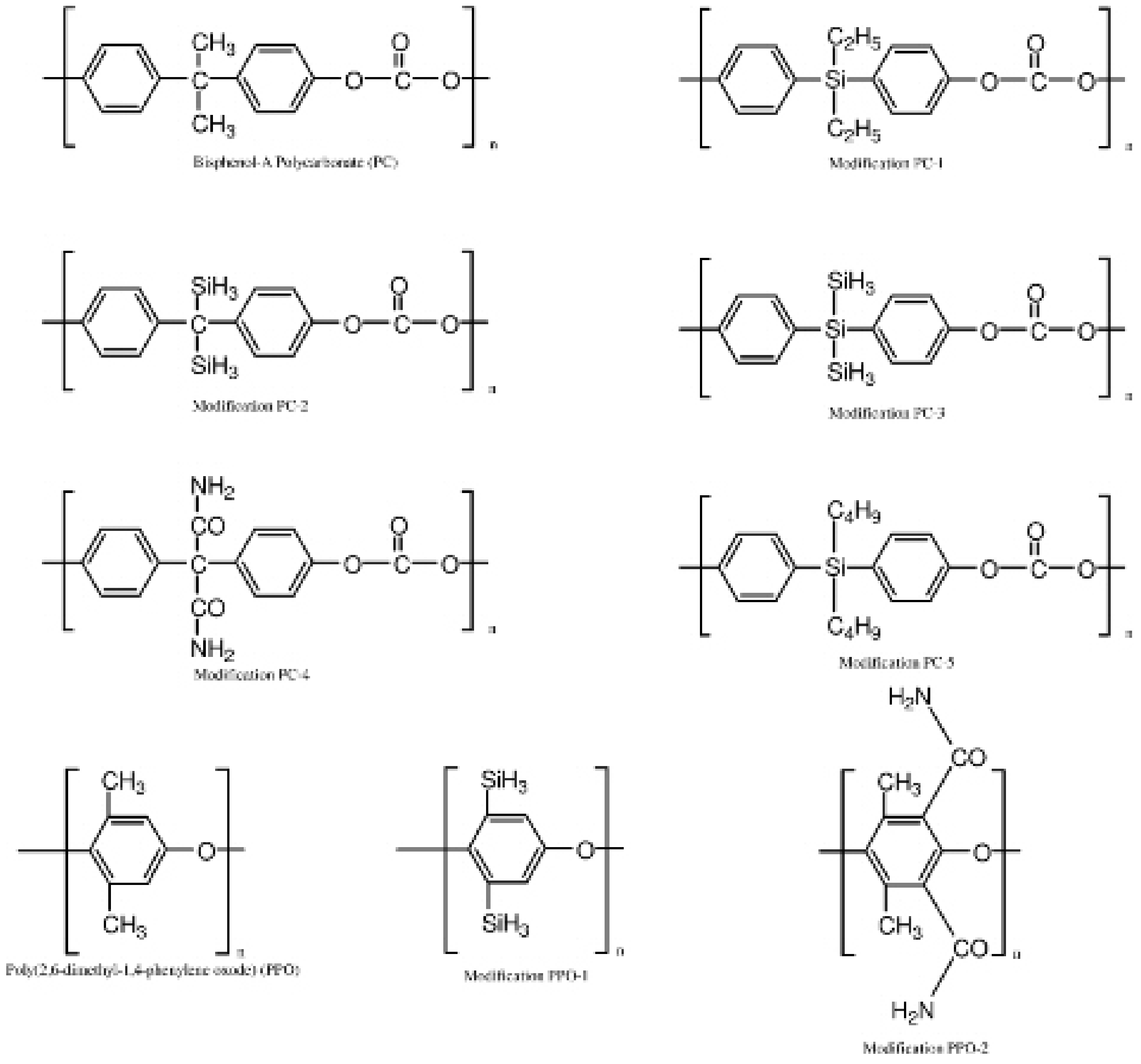

FIGURE 8: Stuctures of the mers of PC and PPO with their modifications predicted to have better mechanical properties.