Neural Comput & Applic (2007) 16:307–316
DOI 10.1007/s00521-007-0090-1

ORIGINAL ARTICLE

Ensemble of hybrid neural network learning approaches for designing pharmaceutical drugs

Ajith Abraham · Crina Grosan · Ştefan Ţigan

Received: 1 December 2006 / Accepted: 21 December 2006 / Published online: 22 February 2007

Abstract Designing drugs is a current problem in the pharmaceutical research. By designing a drug we mean to choose some variables of drug formulation (inputs), for obtaining optimal characteristics of drug (outputs). To solve such a problem we propose an ensemble of three learning algorithms namely an evolutionary artificial neural network, Takagi-Sugeno neuro-fuzzy system and an artificial neural network. The ensemble combination is optimized by a particle swarm optimization algorithm. The experimental data were obtained from the Laboratory of Pharmaceutical Techniques of the Faculty of Pharmacy in Cluj-Napoca, Romania. Bootstrap techniques were used to generate more samples of data since the number of experimental data was low due to the costs and time durations of experimentations. Experiment results indicate that the proposed methods are efficient.

A. Abraham (✉) · C. Grosan
Faculty of Information Technology,
Mathematics and Electrical Engineering,
Norwegian University of Science and Technology,
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
e-mail: ajith.abraham@ieee.org

C. Grosan
Department of Computer Science, Babes-Bolyai University,
Cluj-Napoca 3400, Romania
e-mail: cgrosan@cs.ubbcluj.ro

Ş. Ţigan
Faculty of Medicine, Department of Biostatistics
and Medical Informatics, University Iuliu Hatieganu,
Cluj-Napoca, Romania
e-mail: stigan@umfcluj.ro

Keywords Hybrid learning · Ensemble learning · Evolutionary neural network · Neuro-fuzzy · Drug design

1 Introduction

Drug research and development is comprehensive, expensive, time-consuming and full of risk. It is estimated that a drug from concept to market would take about 12 years and cost more than US$800 million on an average [12]. Several new technologies have hence been developed and applied in drug R&D to shorten the research cycle and to reduce the expenses. The shift of post-genomics towards a systems approach has offered an ever-increasing role for artificial intelligence and robotics. In the postgenomic era, computer-aided drug design (CADD) has considerably extended its range of applications, spanning almost all stages in the drug discovery pipeline, from target identification to lead discovery, from lead optimization to preclinical or clinical trials [24, 29]. Laghaee et al. [6] reviewed some of the latest contributions of AI and robotics to this end and noted the limitations arising from the current independent, exploratory way in which specific solutions are being presented for specific problems without regard to how these could be eventually integrated into one comprehensible integrated intelligent system.

This article presents an ensemble of three learning algorithms namely an evolutionary artificial neural network, Takagi-Sugeno neuro-fuzzy system and an artificial neural network for modeling the situations that interferes the process of designing retard drugs [30, 31]. The ensemble combination is optimized by a particle swarm optimization algorithm.

The problem comes from the pharmaceutical research activity. It refers to a specific class of drugs that has delayed action called generically *retard drugs* [23]. The pharmaceutical experimental situation leads to a mathematical optimization problem. The pharmacist researcher must take into account several variables of formulation of the drug such as: *the speed of mixing turbine, the concentration of the binder, addition speed, the proportion of talc, the proportion of lauril sulfate Na*. We will call these input variables denoted by $X_i, i = \overline{1,n}$. In the problem considered, there are five inputs, X_1, X_2, X_3, X_4 *and* X_5. With those input variables, for each combination of the variables, the researcher obtains a variant of drug with certain characteristics. For each obtained variant, some parameters are measured called *responses* that characterize the drug: *charging performance, average diameter of the pill, Carr value, Hausner value, the flow time* and the *brittleness*. We consider these responses as *outputs* and we formalize them by denoting with $Y_j, j = \overline{1,m}$. For the considered problem there are six outputs denoted by Y_1, Y_2, Y_3, Y_4, Y_5 and Y_6. The costs of experimentations are high and it is necessary to devote a long time to determine all responses for each variant of drug [22]. A snapshot of the experiment data is illustrated in Table 1.

The aim is to determine a combination of variables of formulation $(x_1, x_2, \ldots, x_5)$ such that the responses $(y_1, y_2, \ldots, y_6)$ are optimal. By optimal we mean that outputs respect some conditions. We must take into account some restraints for outputs.

1. The first response, output Y_1, must be maximized, so the goal is to obtain a value as close as possible to 100%.
2. Te second output Y_2 must not outrun some values determined by the fact it is a value representing tablet's diameter. So, the requirement is that $y_2 \in$ [800 μm, 1,000 μm]. A value around the average 900 μm is suitable.
3. The third output Y_3 has also an admissible interval for its value [1, 20], but we must determine it as close as possible to 1.
4. The fourth response, output Y_4, has a narrower interval for its values [1, 1.2], but it also has to be closest to 1.
5. The fifth output Y_5, representing a time quantity, must be as small as possible, but positive.
6. For the last output Y_6, the goal is to minimize it, with positive values, so the 0 value is considered desirable.

The research objective is to search for the values of $X_i i = \overline{1,5}$ for which a drug formulation is obtained with optimal characteristics $Y_j j = \overline{1,6}$. Variables are chosen by the researcher from a continuous domain and not all values are accepted. Accepted variation intervals for inputs, for our problem, are: $X_1 \in [0,50], X_2 \in [1,8], X_3 \in [3,9], X_4 \in [0,5]$, and $X_5 \in [0,1]$.

Computational techniques have proved their efficiency in dealing with problems from medical and pharmaceutical domain. Several computational intelligence techniques (such as neural networks, fuzzy systems, evolutionary algorithms, support vector machines) have been successfully applied for designing different types of drugs. A review of neural networks and genetic algorithms applied for drug design can be found in [17]. A review of genetic algorithms and evolutionary programming for drug design can be found in [1]. Burbidge et al. [21] showed that support vector machines could be very useful for drug design. Artificial neural networks were used for drug design by Tekayama et al. [13], Sun et al. [28] and Hu et al. [16]. Bayesian regularized artificial neural networks were used by Winkler and Burden [8]. A review on natural computing techniques for drug discovery is provided by

Table 1 Sample data showing the *inputs* and *outputs*

Variables of formulation: inputs						Responses: outputs					
Exper. No.	X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
1	20	2	3	5	1	84.0	973.8	4.2	1.043	7.85	1.165
2	40	2	3	0	0	71.9	1150.0	1.6	1.016	8.2	2.264
3	20	8	3	0	1	92.5	1121.4	4.2	1.044	8.83	0.700
4	40	8	3	5	0	88.1	1200.0	3.7	1.038	8.87	1.205
5	20	2	9	5	0	99.2	910.0	5.8	1.061	8.3	1.914
6	40	2	9	0	1	68.2	985.1	4.1	1.043	7.9	2.550
7	20	8	9	0	0	99.1	1010.0	5.3	1.056	9.05	1.160
8	40	8	9	5	1	83.9	925.4	5.5	1.058	8.5	1.265
9	30	5	6	2.5	0.5	85.0	1055.8	3.8	1.036	8.3	1.535
10	30	5	6	2.5	0.5	81.2	1030.0	4.1	1.042	8.37	1.490
11	30	5	6	2.5	0.5	85.0	1060.0	4.1	1.042	8.4	1.535

Solmajer and Zupan [26]. Computational approaches that adopt dynamical models were used by Aradi and Erdi [9].

Esseiva et al. [20] used artificial neural networks (ANNs) to validate illicit drug classification in the profiling method used at the University of Lausanne. Their method established links between samples using a combination of principal component analysis (PCA) and calculation of a correlation value between samples. The application of a neural network was found to be a useful tool to validate the classification of new drug seizures in existing chemical classes.

Kiss and Erdi [27] proposed a novel way of computational modeling by integrating compartmental neural techniques and detailed kinetic description of pharmacological modulation of transmitter–receptor interaction is offered as a method to test the electrophysiological and behavioral effects of potential drugs. Authors also suggested an inverse method as a method for controlling a neural system to realize a prescribed temporal pattern.

Meek et al. [19] reviewed shape signatures, a tool that is effective and easy to use compared with most computer aided drug design techniques. Barat et al. [5] investigated how Monte-Carlo (MC)-based methods are used in the field of drug delivery, indicating what aspects of the complex problems of drug dissolution and design can benefit from this particular approach. Authors examined the existing Direct MC and stochastic cellular automata modeling efforts used to simulate dissolution of pharmaceutical compacts or related phenomena. Stewart et al. [15] proposed the Drug Guru^{TM} which is a web-based computer software program for medicinal chemists that applies a set of transformations, that is, rules, to an input structure. The transformations correspond to medicinal chemistry design rules-of-thumb taken from the historical lore of drug discovery programs. The output of the program is a list of target analogs that can be evaluated for possible future synthesis.

Rest of the paper is organized as follows. In Sect. 2, the computational intelligence techniques used are presented followed by experiment results in Sect. 3. Some conclusions are provided towards the end.

2 Computational intelligence tools (CI)

CI substitutes intensive computation for insight into how complicated systems work. Artificial neural networks, fuzzy inference systems, probabilistic computing, evolutionary computation etc were all shunned by classical system and control theorists. CI provides an excellent framework in unifying them and even by incorporating other revolutionary methods [7].

2.1 Artificial neural networks

Artificial neural networks (ANNs) were designed to mimic the characteristics of the biological neurons in the human brain and nervous system. An artificial neural network creates a model of neurons and the connections between them, and trains it to associate output neurons with input neurons. The network ''learns'' by adjusting the interconnections (called weights) between layers. When the network is adequately trained, it is able to generate relevant output for a set of input data. A valuable property of neural networks is that of generalization, whereby a trained neural network is able to provide a correct matching in the form of output data for a set of previously unseen input data.

Backpropagation (BP) is one of the most famous training algorithms for multilayer perceptrons. Basically, BP is a gradient descent technique to minimize the error E for a particular training pattern. For adjusting the weight (w_k), in the batched mode variant the descent is based on the gradient $\nabla E(\delta E/\delta w_k)$ for the total training set:

$$\Delta w_k(n) = -\varepsilon \frac{\delta E}{\delta w_k} + \alpha \Delta w_k(n-1) \tag{1}$$

The gradient gives the direction of error E. The parameters ε and α are the learning rate and momentum respectively. A good choice of both the parameters is required for training success and speed of the ANN.

In the conjugate gradient algorithm (CGA) a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. A search is made along the conjugate gradient direction to determine the step size, which will minimize the performance function along that line. A line search is performed to determine the optimal distance to move along the current search direction. Then the next search direction is determined so that it is conjugate to previous search direction. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction. An important feature of the CGA is that the minimization performed in one step is not partially undone by the next, as it is the case with gradient descent methods. An important drawback of CGA is the requirement of a line search, which is computationally expensive. Moller [18] introduced the

scaled conjugate gradient algorithm (SCGA) as a way of avoiding the complicated line search procedure of conventional CGA. According to the SCGA, the Hessian matrix is approximated by

$$E''(w_k)p_k = \frac{E'(w_k + \sigma_k p_k) - E'(w_k)}{\sigma_k} + \lambda_k p_k \qquad (2)$$

where E' and E'' are the first and second derivative information of global error function $E(w_k)$. The other terms p_k, σ_k and λ_k represent the weights, search direction, parameter controlling the change in weight for second derivative approximation and parameter for regulating the indefiniteness of the Hessian. In order to get a good quadratic approximation of E, a mechanism to raise and lower λ_k is needed when the Hessian is positive definite. Detailed step-by-step description can be found in the literature [18].

2.2 Evolutionary artificial neural networks (EANN)

The interest in evolutionary search procedures for designing ANN architecture has been growing in recent years as they can evolve towards the optimal architecture without outside interference, thus eliminating the tedious trial and error work of manually finding an optimal network. The advantage of the automatic design over the manual design becomes clearer as the complexity of ANN increases. EANNs provide a general framework for investigating various aspects of simulated evolution and learning. We used the meta-learning evolutionary artificial neural network (MLEANN) framework [3]. Figure 1 illustrates the general interaction mechanism with the learning mechanism of the EANN evolving at the highest level on the slowest time scale. In EANN's evolution can be introduced at various levels. At the lowest level, evolution can be introduced into weight training, where ANN weights are evolved. At the next higher level, evolution can be introduced into neural network architecture adaptation, where the architecture (number of hidden layers, no of hidden neurons and node transfer functions) is evolved. At the highest level, evolution can be introduced into the learning mechanism. A general framework of EANNs which includes the above three levels of evolution is given in Fig. 1.

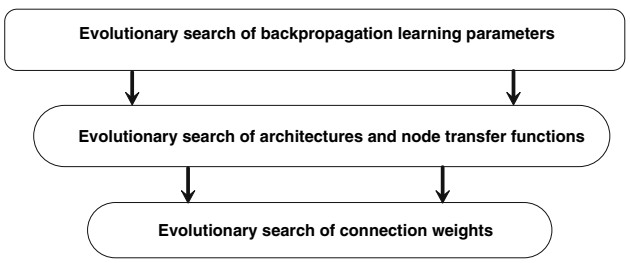

Fig. 1 Interaction of various evolutionary search mechanisms in MLEANN

The efficiency of evolutionary training can be improved significantly by incorporating a local search procedure into the evolution. Evolutionary algorithms are used to first locate a good region in the space and then a local search procedure is used to find a near optimal solution in this region. It is interesting to consider finding good initial weights as locating a good region in the space. Defining that the basin of attraction of a local minimum is composed of all the points, sets of weights in this case, which can converge to the local minimum through a local search algorithm, then a global minimum can easily be found by the local search algorithm if the evolutionary algorithm can locate any point, i.e., a set of initial weights, in the basin of attraction of the global minimum. BP algorithm is used as the local search algorithm. All the randomly generated architectures of the initial population are trained by BP algorithm for a fixed number of epochs. The learning rate and momentum of the BP algorithm are adapted according to the problem. The basic algorithm of the proposed EANN framework is given below.

1. Set $t = 0$ and randomly generate an initial population of neural networks with architectures, node transfer functions and connection weights assigned at random.
2. Evaluate fitness of each ANN using BP algorithm
3. Based on fitness value, select parents for reproduction
4. Apply mutation to the parents and produce offspring (s) for next generation. Refill the population back to the defined size.
5. Repeat step 2
6. STOP when the required solution is found or number of iterations has reached the required limit.

Architecture of the chromosome is depicted in Fig. 2. We used a special mutation operator, which decreases the mutation rate as the algorithm greedily proceeds in the search space. If the allelic value x_i of the ith gene ranges over the domain a_i and b_i the mutated gene x_i' is drawn randomly uniformly from the interval $[a_i, b_i]$.

$$x_i' = \begin{cases} x_i + \Delta(t, b_i - x_i), & \text{if } \omega = 0 \\ x_i + \Delta(t, x_i - a_i), & \text{if } \omega = 1 \end{cases} \qquad (3)$$

where ω represents an unbiased coin flip $p(\omega = 0) = p(\omega = 1) = 0.5$, and

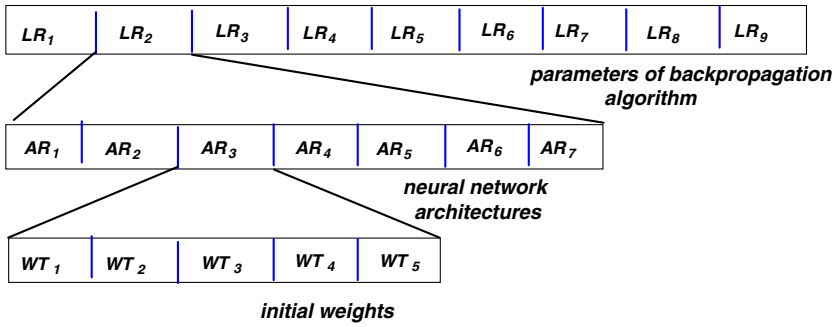

Fig. 2 Chromosome representation of MLEANN

$$\Delta(t,x) = x\left(1 - \gamma^{\left(1-\frac{t}{t_{\max}}\right)^{b}}\right) \tag{4}$$

defines the mutation step, where γ is the random number from the interval $[0,1]$ and t is the current generation and $t_{\max}$ is the maximum number of generations. The function Δ computes a value in the range $[0,x]$ such that the probability of returning a number close to zero increases as the algorithm proceeds with the search. The parameter b determines the impact of time on the probability distribution Δ over $[0,x]$. Large values of b decrease the likelihood of large mutations in a small number of generations.

2.3 Neuro-fuzzy computing

Neuro-fuzzy (NF) computing is a popular framework for solving complex problems [4, 11]. If we have knowledge expressed in the form of linguistic rules, we can build a fuzzy inference system (FIS), and if we have data, or can learn from a simulation (training) then we can use ANNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be advantage from the viewpoint of ANN. We used the adaptive neuro fuzzy inference system (ANFIS) implementing a Takagi-Sugeno type FIS. We modified the ANFIS model to accommodate the multiple outputs [11]. Figure 3 depicts the six-layered architecture of multiple output ANFIS and the functionality of each layer is described below. Readers are advised to consult [11] for technical more details of ANFIS.

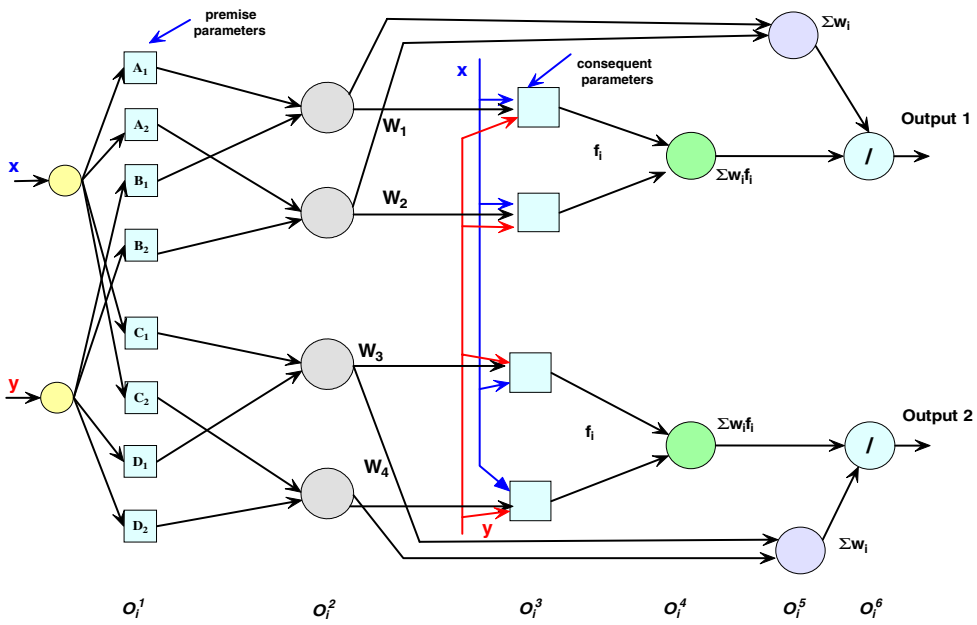

Fig. 3 Architecture of ANFIS with multiple outputs

Layer 1. Every node in this layer has a node function. $O_i^1 = \mu_{A_i}(x)$, for $i = 1, 2$ or $O_i^1 = \mu_{B_{i-2}}(y)$, for $i = 3, 4, \ldots$. O_i^1 is the membership grade of a fuzzy set A($= A_1, A_2, B_1$ or B_2) and it specifies the degree to which the given input x (or y) satisfies the quantifier A. Usually the node function can be any parameterized function. A Gaussian membership function is specified by two parameters c (membership function center) and σ (membership function width). Gaussian $(x, c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}$. Parameters in this layer are referred to premise parameters.

Layer 2. Every node in this layer multiplies the incoming signals and sends the product out. Each node output represents the firing strength of a rule. $O_i^2 = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1, 2, \ldots,$ In general any T-norm operator that performs fuzzy ''AND'' can be used as the node function in this layer.

Layer 3. The rule consequent parameters are determined in this layer. $O_i^3 = f_i = xp_i + yq_i + r_I$, where $\{p_i, q_i, r_i\}$ are the rule consequent parameters.

Layer 4. Every node i in this layer is with a node function $O_i^4 = \sum \overline{w_i} f_i = \sum \overline{w_i}(p_i x + q_i y + r_i)$, where $\overline{w_i}$ is the output of layer 2

Layer 5. Every node in this layer aggregates all the firing strengths of rules $O_i^5 = \sum \overline{w_i}$.

Layer 6. Every ith node in this layer calculates the individual outputs.

$$O_i^6 = \overline{\text{Output}} = \frac{\sum \overline{w_i} f_i}{\sum_i \overline{w_i}}, \quad i = 1, 2 \ldots.$$

ANFIS makes use of a mixture of backpropagation to learn the premise parameters and least mean square estimation to determine the consequent parameters. A step in the learning procedure has two parts: In the first part the input patterns are propagated, and the optimal conclusion parameters are estimated by an iterative least mean square procedure, while the antecedent parameters (membership functions) are assumed to be fixed for the current cycle through the training set. In the second part the patterns are propagated again, and in this epoch, backpropagation is used to modify the antecedent parameters, while the conclusion parameters remain fixed. This procedure is then iterated.

2.4 Ensemble of intelligent paradigms using particle swarm optimization

Ensemble Learning combines multiple learned models under the assumption that the combined model would perform better than a direct stand alone approach. The decisions of multiple hypotheses are combined in ensemble learning to produce more accurate results. The problem becomes more complicated when we have to optimize several performance (error) measures. For the drug design problem, the task is to optimize the root mean squared error (RMSE) and correlation coefficient (CC).

The first step is to carefully construct the different models (neural network, neuro-fuzzy model and evolutionary neural network) to achieve the best generalization performance. Test data is then passed through these individual models and the corresponding outputs are recorded. Suppose the output values predicted by ANN, EANN and NF are a_n, b_n and c_n, respectively, and the corresponding desired value is x_n. Our task is to combine a_n, b_n *a*nd c_n so as to get the best output value (x_n) that maximizes the CC and minimizes the RMSE values.

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed inspired by social behavior of bird flocking or fish schooling. Initially, a population of particles is randomly generated [14]. Each particle represents a potential solution and has a position represented by a position vector x_i. A swarm of particles moves through the problem space, with the moving velocity of each particle represented by a velocity vector v_i. At each time step, a function f_i—representing a quality measure—is calculated by using x_i as input. Each particle keeps track of its own *best* position, which is associated with the best fitness it has achieved so far in a vector p_i. Furthermore, the *best* position among all the particles obtained so far in the population is kept track of as p_g. In addition to this global version, another version of PSO keeps track of the *best* position among all the topological neighbors of a particle. At each time step t, by using the individual best position, p_i (t), and the global best position, p_g (t), a new velocity for particle i is updated by

$$v_i(t+1) = v_i(t) + c_1\phi_1(p_i(t) - x_i(t)) + c_2\phi_2(p_g(t) - x_i(t)) \tag{5}$$

where c_1 and c_2 are positive constants and φ_1 and φ_2 are uniformly distributed random numbers in [0,1]. The term v_i is limited to the range of $\pm V_{\max}$ (if the velocity violates this limit, it is set to its proper limit). Changing velocity this way enables the particle i to search around both its individual best position, p_i, and global best position, p_g. Based on the updated velocities, each particle changes its position according to:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{6}$$

Using a PSO algorithm a search for the optimal values for the linear parameters *m, n*, and *o* are performed such that

$$m+n+o=1 \text{ and } a_n \times m + b_n \times n + c_n \times o \approx x_n \quad (7)$$

so as to minimize RMSE and maximizes the CC. The ensemble model is illustrated in Fig. 4. The objective function is modeled as

$$\text{Minimize}(Z) = \text{RMSE} + 1 - \text{CC} \quad (8)$$

3 Experiment setup and results

3.1 Data preparation

Very often, for pharmaceutical drug design problems the relation between inputs and outputs is unknown and the data are difficult to obtain due to costs and time constraints. Sometimes the sample data is too poor and the regression functions are not suitable to describe the association between variables. We resort, in this case, to a re-sampling method that allow us to manage uncertainty. The aim is to improve our sample of data with pseudo data and to evaluate some statistical parameters. We used a bootstrap method of re-sampling data [2, 10, 25]. We grouped each input variable of formulation X_i, $i = \overline{1,5}$ with each output $Y_j, j = \overline{1,6}$. Finally, among the bootstrap simulated sets of X_i variables we select a combination of inputs, a set of values $x_i, i = \overline{1,5}$, that correspond to best situated values of Y_j.

The experiment system consists of two stages: building and modeling the different models described in Sect. 2 and performance evaluation. Experiments were repeated three times and the worst errors are reported. The test data is passed through the trained network to evaluate the learning efficiency of the considered models.

3.2 Simulation of models

Our preliminary experiments helped us to formulate a feedforward neural network with 1 input layer, 1 hidden layer and an output layer [5–10–6]. Input layer consists of five neurons corresponding to the input variables. The hidden layer consists of 10 neurons using tanh-sigmoidal activation function. Training was terminated after 2,000 epochs and we achieved a RMSE of 0.0362. Figure 5 shows the convergence of SCGA during the 2,000 epochs training.

We have applied the MLEANN algorithm for drug design. For performance comparison, we used the same set of training and test data that were used for experimentations with conventional design of neural net-

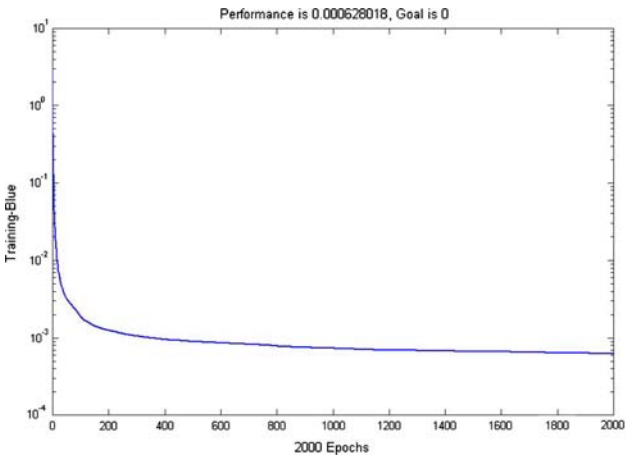

Fig. 5 Convergence of SCGA training

Table 2 Parameters used for evolutionary design of artificial neural networks

Population size	30
Maximum no of generations	25
Number of hidden nodes	5–9 hidden nodes
Activation functions	tanh (*T*), logistic (*L*), sigmoidal (*S*), tanh-sigmoidal (*T**), log-sigmoidal (*L**)
Output neuron	Linear
Training epochs	500
Initialization of weights	±0.1
Ranked based selection	0.50
Learning rate	0.15–0.01
Momentum	0.15–0.01
Elitism	5%
Initial mutation rate	0.70

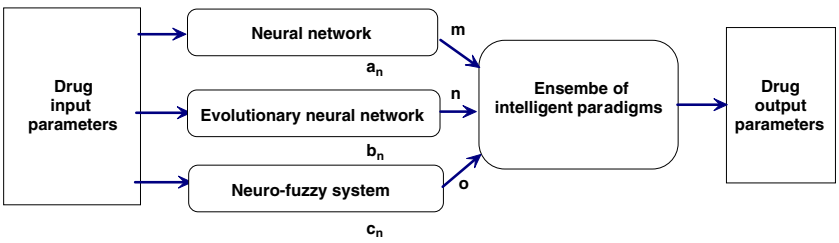

Fig. 4 Architecture of the ensemble model

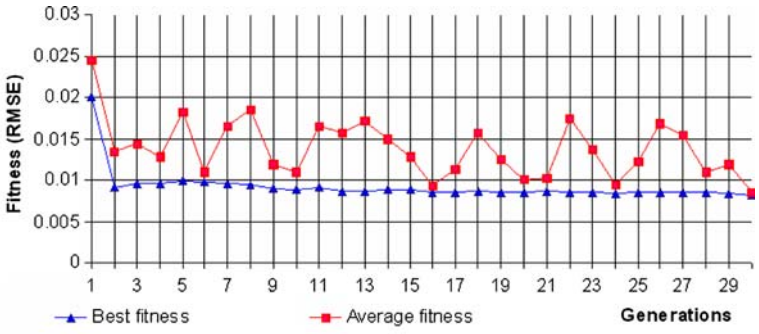

Fig. 6 Convergence of MLEANN algorithm

works and the neuro-fuzzy system. Fitness value is calculated based on the RMSE achieved on the test set. We have considered the best-evolved neural network as the best individual of the last generation. All the genotypes were represented using real coding and the initial populations were randomly generated based on the parameters shown in Table 2. MLEANN learning (convergence) showing the best fitness values is illustrated in Fig. 6. Compared to a direct neural network approach, the MLEANN approach resulted in better performance measures with only five hidden neurons.

For training the neuro-fuzzy (NF) model, we used two Gaussian membership functions for each input variables and 64 rules were learned using the hybrid training method. Training was terminated after 30 epochs. Figures 7, 8, 9, 10 depict some sample input–output surface for the developed fuzzy if–then rule base.

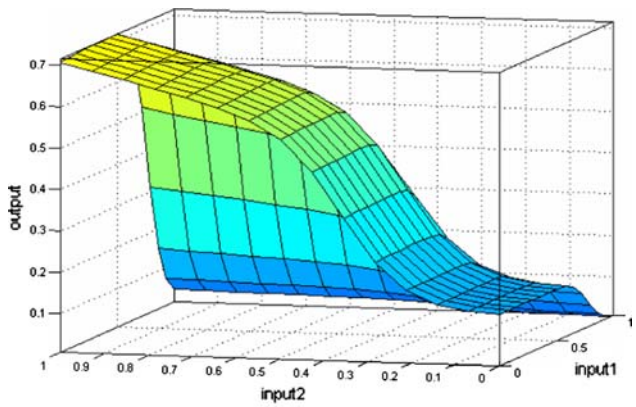

Fig. 7 Learned surface between X_1, X_2 and Y_1

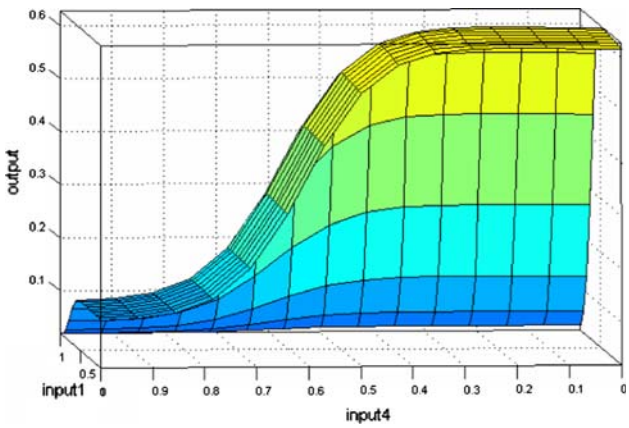

Fig. 9 Learned surface between X_1, X_4 and Y_1

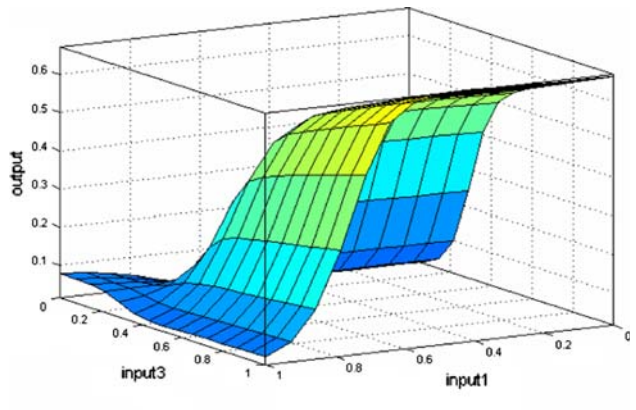

Fig. 8 Learned surface between X_1, X_3 and Y_1

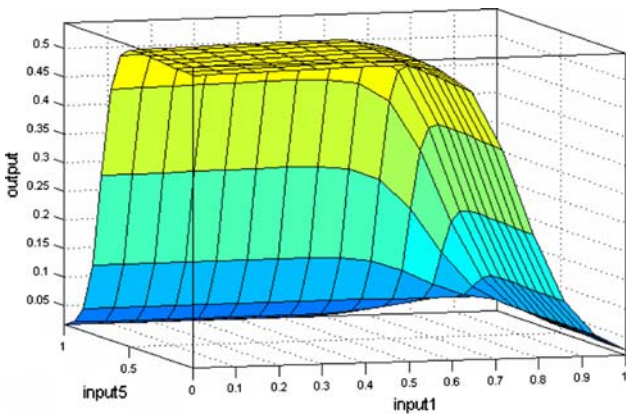

Fig. 10 Learned surface between X_1, X_5 and Y_1

Table 3 Test results and performance comparison of drug design system

Output	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
Ensemble approach						
RMSE	0.00924	0.01017	0.00821	0.00891	0.00792	0.00834
CC	0.996	0.994	0.994	0.993	0.993	0.997
Evolutionary artificial neural networks						
RMSE	0.011	0.019	0.0123	0.0129	0.0132	0.0101
CC	0.990	0.993	0.994	0.990	0.989	0.998
Artificial neural networks						
RMSE	0.0487	0.039	0.043	0.040	0.038	0.039
CC	0.979	0.968	0.967	0.956	0.966	0.975
MIMO neuro-fuzzy system						
RMSE	0.0187	0.0213	0.0134	0.0145	0.0190	0.0132
CC	0.992	0.989	0.993	0.987	0.991	0.990

Table 3 summarizes the performances of MLE-ANN, neuro-fuzzy system and neural network for the test data showing the individual RMSE and CC for outputs Y_1, Y_2, Y_3, Y_4, Y_5 and Y_6. For the ensemble learning a swarm size of 20 was used and c_1 and c_2 were set = 1.49. Empirical results using the ensemble method are depicted in Table 3.

4 Conclusions

In this paper, we have proposed an ensemble of three learning algorithms for drug design. Test results reveal that the proposed ensemble model is capable of modeling all the outputs accurately when compared to the individual approaches. Compared to artificial neural network, neuro-fuzzy system performed better in terms of RMSE and correlation coefficient. Another important advantage of neuro-fuzzy system is the interpretability of the results using *if–then* rules. The proposed intelligent system might be useful for drug design researchers, companies engaged in drug business etc. Performance could have been improved by providing more training data. The most important achievement of this result is that it gives the researcher a new starting point of experimentation instead of doing another 20–30 experiments and to arrive at the same conclusion as the ensemble model recommends. Our intention is to create some flexible and adaptable tools for modeling and simulation using numerical and statistical tools, all this with computer support and distributed on Internet.

Acknowledgments Authors would also like to thank the colleagues of the Department of Maxillofacial Surgery, University of Medicine and Pharmacy, Iuliu Hatieganu Cluj-Napoca, for the initial contributions of this research

References

1. Parrill AL (1996) Evolutionary and genetic methods in drug design. Drug Discov Today 1(12):514–521
2. Zoubir AM, Iskander DR (1998) Bootstrap MATLAB Toolbox. Software reference manual
3. Abraham A (2004) Meta-learning evolutionary artificial neural networks. Neurocomp J 56c:1–38
4. Abraham A (2001) Neuro-fuzzy systems: state-of-the-art modeling techniques, connectionist models of neurons, learning processes, and artificial intelligence. In: Mira J, Prieto A (eds.) Lecture notes in computer science. Springer, Berlin, LNCS 2084, pp 269–276
5. Barat A, Ruskin HJ, Crane M (2006) Probabilistic models for drug dissolution. Part 1. Review of Monte Carlo and stochastic cellular automata approaches. Simul Model Pract Theory 14(7):843–856
6. Laghaee A, Malcolm C, Hallam J, Ghazal P (2005) Artificial intelligence and robotics in high throughput post-genomics. Drug Discov Today 10(18):1253–1259
7. Carlsson C, Fullér R (1998) Multiobjective optimization with linguistic variables. In: Proceedings of the sixth European congress on intelligent techniques and soft computing, Aachen, September 7–10, 1998, Verlag Mainz
8. Winkler DA, Burden FR (2004) Bayesian neural nets for modeling in drug discovery. Drug Discov Today: BIOSILICO 2(3):104–111
9. Aradi I, Erdi P (2006) Computational neuropharmacology: dynamical approaches in drug discovery. Trends Pharmacol Sci 27(5):240–243
10. Carpenter J, Goldstein H, Rasbash J (1999) A non-parametric bootstrap for multilevel models. Multilevel Model Newsl 11:2–5
11. Jang SR, Sun CT, Mizutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Englewood Cliffs
12. DiMasi JA, Hansen RW, Grabowski HG (2003) The price of innovation: new estimates of drug development costs. J Health Econ 22:151–185
13. Takayama K, Fujikawa M, Obata Y, Morishita M (2003) Neural network based optimization of drug formulations. Adv Drug Deliv Rev 55(9):1217–1231
14. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, Vol. IV, pp 1942–1948. IEEE service center, Piscataway, NJ

15. Stewart KD, Shiroda M, James CA (2006) Drug Guru: a computer software program for drug design using medicinal chemistry rules. Bioorg Med Chem 14(20):7011–7022
16. Hu L, Chen GH, Chau RMW (2006) A neural networks-based drug discovery approach and its application for designing aldose reductase inhibitors. J Mol Graph Model 24(4):244–253
17. Teroth L, Gasteiger J (2001) Neural networks and genetic algorithms in drug design. Drug Discov Today 6(2):102–108
18. Moller AF (1993) A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw 6:525–533
19. Meek PJ, Liu Z, Tian L, Wang CY, Welsh WJ, Zauhar RJ (2006) Shape signatures: speeding up computer aided drug discovery. Drug Discov Today 11(19–20):895–904
20. Esseiva P, Anglada F, Dujourdy L, Taroni F, Margot P, Pasquier ED, Dawson M, Roux C, Doble P (2005) Chemical profiling and classification of illicit heroin by principal component analysis, calculation of inter sample correlation and artificial neural networks. Talanta 67(2):360–367
21. Burbidge R, Trotter M, Buxton B, Holden S (2001) Drug design by machine learning: support vector machines for pharmaceutical data analysis. Comput Chem 26(1):5–14
22. Câmpean R, Prodan A (2003) Biomatematică – aplicatii în Excel, Editura Medicală Universitară ''Iuliu Hatieganu'', Cluj-Napoca, ISBN: 973-693-016-5
23. Câmpean R, Prodan A (2003) A rating model applied for designing drugs. In: Proceedings of the 12-th IASTED international conference on applied simulation and modelling, Marbella, Spain, pp 557–561, ACTA press, ISBN: 0-88986-384-9, ISSN: 1021–8181
24. Agatonovic-Kustrin S, Beresford R (2000) Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. J Pharm Biomed Anal 22(5):717–727
25. Hesterberg T, Monaghan S, Moore DS, Clipson A, Epstein R (2003) Bootstrap methods and permutation tests. W. H. Freeman and Company, New York
25. Solmajer T, Zupan J (2004) Optimization algorithms and natural computing in drug discovery. Drug Discov Today: Technol 1(3):247–252
27. Kiss T, Érdi P (2006) From electric patterns to drugs: perspectives of computational neuroscience in drug design. Biosystems 86(1–3):46–52
28. Sun Y, Peng Y, Chen Y, Shukla AJ (2003) Application of artificial neural networks in the design of controlled release drug delivery systems. Adv Drug Deliv Rev 55(9):1201–1215
29. Tang Y, Zhu W, Chen K, Jiang H (2006) New technologies in computer-aided drug design: toward target identification and new chemical entity discovery. Drug Discov Today: Technol 3(3):307–313
30. Grosan C, Abraham A, Tigan S (2006) Engineering drug design using a multi-input multi-output neuro-fuzzy system, 8th International symposium on symbolic and numeric algorithms for scientific computing (SYNASC'06), Timisoara, Romania, IEEE CS Press, pp 365–371
31. Grosan C, Abraham A, Tigan S, Chang T-G, Kim DH (2006) Evolving neural networks for pharmaceutical research, International conference on hybrid information technology (ICHIT'06), IEEE Press, Korea, pp 13–19