Neural Processing Letters **16:** 187–199, 2002.

Associative Neural Network

IGOR V. TETKO★
Institute for Bioinformatics, MIPS, GSF, Ingolstädter Landstraße 1, D-85764 Neuherberg, Germany and Biomedical Department, Institute of Bioorganic and Petroleum Chemistry, Ukrainian Academy of Sciences, Murmanskaya 1, Kiev-660, 253660, Ukraine.
e-mail: itetko@vcclab.org

Abstract. An associative neural network (ASNN) is a combination of an ensemble of the feed-forward neural networks and the K-nearest neighbor technique. The introduced network uses correlation between ensemble responses as a measure of distance among the analyzed cases for the nearest neighbor technique and provides an improved prediction by the bias correction of the neural network ensemble both for function approximation and classification. Actually, the proposed method corrects a bias of a global model for a considered data case by analyzing the biases of its nearest neighbors determined in the space of calculated models. An associative neural network has a memory that can coincide with the training set. If new data become available the network can provide a reasonable approximation of such data without a need to retrain the neural network ensemble. Applications of ASNN for prediction of lipophilicity of chemical compounds and classification of UCI letter and satellite data set are presented. The developed algorithm is available on-line at http://www.virtuallaboratory.org/lab/asnn.

Key words. associative memory, bias correction, classification, function approximation, *k*-nearest neighbors, memory-based methods, memoryless, prototype selection

1. Introduction

The traditional multi-layer neural network (MLP) is a memoryless approach. This means that after training is complete all information about the input patterns is stored in the neural network weights and input data are no longer needed, i.e. there is no explicit storage of any presented example in the system. Contrary to that, such methods as the k-nearest-neighbors (KNN) (e.g., [1]), the Parzen-window regression (e.g., [2]), etc. represent the memory-based approaches. These approaches keep in memory the entire database of examples and their predictions are based on some local approximation of the stored examples. The neural networks can be considered global models, while the other two approaches are usually considered local models [3].

Consider a problem of multivariate function approximation from examples, i.e. finding a mapping $R^m => R^n$ from a given set of sampling points. For simplicity, let us assume that $n = 1$. A global model provides a good approximation of the global metric of the input data space R^m. However, if the analyzed function, f, is too

★Address for correspondence: Dr. Igor Tetko, Institute for Bioinformatics, GSF - Forschungszentrum für Umwelt und Gesundheit, GmbH Ingolstädter Landstraße 1, D-85764 Neuherberg, Germany.

complicated, there is no guarantee that all details of f, i.e. its fine structure, will be represented. Thus, the global model can be inadequate because it does not describe equally well the entire state space with poor performance of the method being mainly due to a high bias of the global model in some particular regions of space. The same problem of bias is also pertinent if neural networks are used for classification. The MLP variance can also contribute to poor performance of this method [4]. However, the variance can be decreased by analyzing a large number of networks, i.e. using artificial neural network ensemble, and taking, for example, a simple average of all networks as the final model. The problem of bias of MLP cannot be so easily addressed simply by using larger neural networks since such networks can fall in a local minimum and thus can still have a considerable bias. Thus, one of the motivation for this article is to provide a method that can estimate and correct bias of neural networks for both regression and classification.

Sometimes in practical applications a user can be interested in analyzing new data that will require extrapolation, or data that have some changes in their basic functional properties compared to the training set. In both such cases we assume that the user have some new data ('fresh' data) covering domains of his interest and such data can be used to improve the existing model. One possibility to improve prediction ability of neural networks for the user's data is to retrain MLPs including the fresh data or to develop a model using exclusively the fresh data. Unfortunately, development of neural networks is a time-consuming task and in some cases such calculations could not be performed due to practical limitations of time, resources or privacy issues. It is also possible that the amount of the fresh data could be very small and insufficient to completely develop a new model. Thus, it will be important to have a method that can improve performance of MLP for the prediction of data that are similar to the user's data, without a need to completely retrain the neural networks. It will be shown that the proposed method provides a powerful solution to these problems.

The outline of the Letter is as follows. In first, I will briefly describe traditional methods, such as KNN and ensemble of neural networks, and then introduce a new method that is a combination of both these approaches. Two first examples, sine function interpolation and classification of UCI data sets, will be used to show an importance of bias correction for the improvement of neural network performance in regression and classification studies. The three further examples, sine, Gauss function and identity function extrapolation will be used to demonstrate properties of new method for prediction of data that have some features changed compared to the training set. The last example will demonstrate an industrial application of lipophilicity prediction, that is an important physico-chemical parameter of molecules.

2. Methods

Let us consider a training set consisting of N input-output pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, ((\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i$ is a vector of input parameters. Our purpose is to predict y-value of a new data case $\mathbf{x}$.

2.1. DESCRIPTION OF THE KNN METHOD

KNN method was used as

$$z(\mathbf{x}) = \frac{1}{k} \sum_{j \epsilon N_k(\mathbf{x})} y(\mathbf{x}_j) \tag{1}$$

where $z(\mathbf{x})$ is a predicted value for case $\mathbf{x}$, $N_k(\mathbf{x})$ is the collection of the k nearest neighbors of $\mathbf{x}$ among the input vectors in the training set $\{\mathbf{x}_i\}_{i=1}^{N}$ using the Euclidian metric $\mathrm{d}(\mathbf{x}, \mathbf{xi}) = \|\mathbf{x} - \mathbf{x}_\mathrm{i}\|$.

2.2. DESCRIPTION OF NEURAL NETWORK

In the current study ensemble of $M = 100$ neural networks, if not mentioned otherwise, was trained according to Levenberg-Marquardt algorithm [5]. Each neural network had the same number of hidden neurons in one input hidden layer. The input and output values were normalized to (0.1,0.9) interval and the sigmoid activation function was used for all neurons. Half of the data cases were selected by chance and were used as a training set for each neural network [6]. The remaining cases were used as a validation set in the early stopping method [7]. Thus, each neural network had its own training and validation sets. Following ensemble learning a simple average of all networks was used to predict the test patterns.

2.3. DESCRIPTION OF THE ASSOCIATIVE NEURAL NETWORK

Consider an ensemble of M neural networks

$$[\mathbf{ANNE}]_M = \begin{bmatrix} ANN_1 \\ \vdots \\ ANN_j \\ \vdots \\ ANN_M \end{bmatrix} \tag{2}$$

The prediction of a case $\mathbf{x}_i, i = 1, \ldots, N$ can be represented by a vector of output values $\mathbf{z}_i = \{z_j^i\}_{j=1}^{M}$ where $j = 1, \ldots, M$ is the index of the network within the ensemble.

$$[\mathbf{x}_i] \bullet [\mathbf{ANNE}]_M = [\mathbf{z}_i] = \begin{bmatrix} z_1^i \\ \vdots \\ z_j^i \\ \vdots \\ z_M^i \end{bmatrix} \tag{3}$$

As it was already mentioned, a simple average $\bar{z}_i = \frac{1}{M} \sum_{j=1,M} z_j^i$ was used in this study to predict the test cases with MLPs.

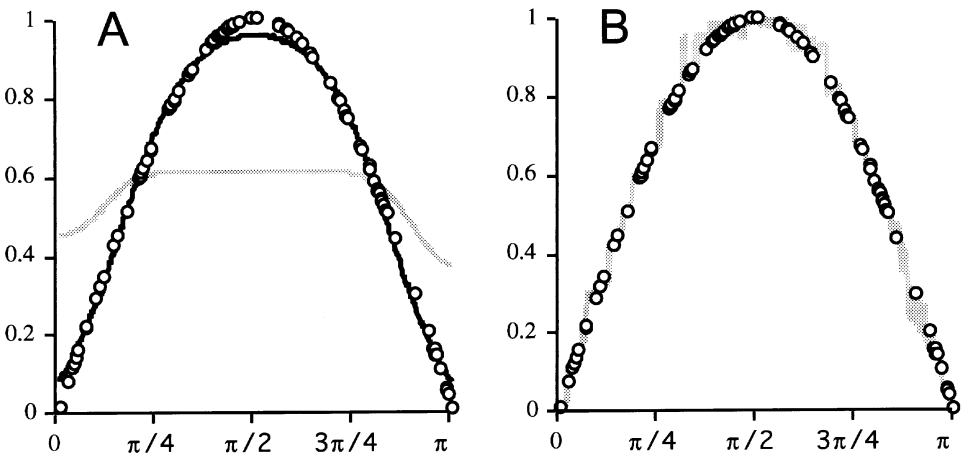

Figure 1. A) Sine function $y = \sin(x = x_1 + x_2)$ interpolation by neural networks with one (gray line) and two hidden neurons (black line) trained using 100 cases (circles). B) ASNN results calculated with one hidden neuron (gray line) for the same example.

In our previous studies [8] we proposed to use the square of linear Person's correlation coefficient [9] between the vectors of predicted values $\mathbf{z}_i$ and $\mathbf{z}_j$ as a measure of similarity (proximity) of analyzed cases in the space of models. Other similarity measures, e.g. Spearman or Kendall's non-parametric rank correlation could also be used [17]. The current study was done with the Spearman non-parametric coefficient, r_{ij}. For negative values of this coefficient $r_{ij} < 0$ a zero correlation $r_{ij} = 0$ was used.

In some regions of data space the ensemble predictions $\bar{z}_i$ could have significant bias. Such biases can be easily seen for neural network predictions at Figure 1A. For example, all cases with $x_1 + x_2 = \pi/2$ have their predicted y-values that are below the target value of the sine function $y = 1$. In order to improve neural network performance in such regions of space the ensemble predictions $\bar{z}_i$ were corrected according to formula

$$\bar{z}'_i = \bar{z}_i + \tfrac{1}{k} \sum_{j \in N_k(\mathbf{x})} (y_j - \bar{z}_j) \tag{4}$$

where y_i are the experimental values, $N_k(\mathbf{x})$ is the collection of the k nearest neighbors of $\mathbf{x}$ among the input vectors in the training set $\{\mathbf{x}_i\}_{i=1}^N$ determined using Spearman non-parametric rank correlation coefficient r_{ij}, as described above. Since the variance of ensemble prediction $\bar{z}_i$ can be made small by increasing a number of neural networks in the ensemble, the difference $(y_i - \bar{z}_i)$ mainly corresponded to the bias of the neural network ensemble for the case $\mathbf{x}_i$. Thus this formula explicitly corrected the bias of the analyzed case according to the observed biases calculated for the neighboring cases.

I refer to the proposed method as associative neural network (ASNN), since the final prediction of new data is done according to the cases, i.e. prototypes of the ana-

lyzed example or associations, found in the 'memory' of the neural network. In the considered example memory of ASNN is equal to the input training set. However, there is no requirement that the memory should always coincide with the training set. For example, if some new data become available to the user these data could be used as the memory of neural networks. This provides a possibility to improve neural network results for function extrapolation without a need to retrain their weights.

3. Data Sets

An example of the sine function

$$y = \sin(\mathbf{x}) \tag{5}$$

with dimension of vector $\mathbf{x}$ equal to 1 and 2 (such as $x = x_1 + x_2$) was used to demonstrate bias correction and extrapolation of neural networks for the function approximation. The training and test sets included $N = 100$ and 1000 cases, respectively, and the input values were uniformly distributed over the interval $(0,\pi)$.

An extrapolation example for classification problem was developed according to Marcus [10]. In this experiment strings of six binary digits were presented to an auto-associate MLP trained to perform identity function. The networks had 5 hidden neurons and were trained on the 32 binary strings representing even numbers in the range 0 . . . 64 [i.e., 0 0 0 0 0 0 . . . 1 1 1 1 1 0]. The test set (extrapolation) consisted of 32 odd numbers in the range 1 . . . 63 [i.e., 0 0 0 0 0 1. . . 1 1 1 1 1 1]. Marcus found that the network did not extrapolate the identity function to odd numbers. Instead, the network would respond incorrectly, for example it would typically respond to the input [1 1 1 1 1 1] with the output [1 1 1 1 1 0].

The performance of developed method for real data was analyzed using letters and satellite data sets from the UCI Machine Learning databases [11] and a program to predict lipophilicity of chemical compounds [12].

4. Results

4.1. SINE FUNCTION INTERPOLATION

This example was used to demonstrate properties of the proposed method for function approximation. The first analysis was performed using dimension of vector $\mathbf{x}$ equal to 1. The number $k = 1$ was selected to provide minimum leave-one-out error (LOO) for the training set using KNN. This method calculated the root mean squared error, rms = 0.019, for the test set. A similar result, rms = 0.022, was calculated by an ensemble of neural networks with 2 hidden neurons. The neural networks had a larger bias and calculated a lower prediction ability, rms = 0.24, when only one hidden neuron was used. However, for both numbers of hidden neurons neural networks clearly underlined a significant bias and poor performance of this method near maximum, $x = \pi/2$, and tails, $x = 0$ and $x = \pi$, of the function.

The performance of nearest neighbors was about an order of magnitude lower, rms = 0.16, $k = 2$, if two-dimension input data were used. Thus, the Euclidian metric used was not optimal for the KNN method and this method was unable to correctly determine the nearest neighbors in the space of such variables. On the contrary, the neural networks with one and two hidden neurons both provided results that were very similar to the analysis using one-dimensional example, with rms = 0.26 and rms = 0.025, respectively (Figure 1(A)). Thus, both types of networks correctly learned the internal metric of the example, i.e. $x = x_1 + x_2$.

A use of ASNN given by (4) provided an improvement of the results of neural networks, and rms = 0.025 and rms = 0.008 were calculated following analysis of networks with one and two hidden neurons respectively (Figure 1(B)). The proposed method improved performance of traditional neural networks in the regions were they had significant bias. An increase of the number of hidden neurons further improved performance of both methods. For example, with 5 hidden neurons rms = 0.012 and rms = 0.005 were calculated by MLPs and ASNN, respectively. Notice that a straightforward use of the nearest neighbor method in the space of neural network models, i.e. if we discard $\bar{z}_i$ and $\bar{z}_j$ terms in (4), could not improve neural network results beyond those calculated by the KNN in the original Euclidian space, i.e. rms = 0.033.

This example demonstrated that the MLPs provided a basic mapping of the input space (i.e., they learned the physical metric of this space $x = x_1 + x_2$) while the use of (4) provided a detailed adjustment of neural networks responses by a correction of their biases. The prediction ability of the ASNN was improved in comparison to both nearest neighbors and traditional neural networks.

4.2. CLASSIFICATION OF UCI DATA SETS

The classification tasks are traditionally used to compare different neural network training algorithms. Two such data sets, 'letters' and 'satellite', from the UCI Machine Learning databases [11] were used in the current study. The letters data set contains 16000 training and 4000 test patterns, 16 input features and 26 classes (A-Z) of machine-printed characters from 20 different fonts. The best result, 1.5% errors, for the test set was calculated by Schwenk and Bengio [9] (Table I). The authors achieved this result after careful study of different strategies of AdaBoost

Table I. Test error rates on the UCI data sets

Data set	CART boosted[1]	C4.5 boosted[2]	MLP boosted[3]	ASNN
letter	3.4%	3.3%	1.5%	1.8%
sattelite	8.8%	8.9%	8.1%	7.8%

[1]results from [26];
[2]results from [27];
[3]results from [9].

algorithm used to improve performance of neural networks. They applied quite large neural network with two hidden layers, 16-70-50-26.

The cascade-correlation architecture [14] neural networks (100 networks in ensemble) with a maximum number of 50 hidden units (i.e., $16 \times 50 \times 26$ architecture) were used in the current study. The training was performed with efficient partition algorithm as described elsewhere [13] and 4.1% mistakes were calculated for the test set. The use of the ASNN improved this result and only 1.8% mistakes were calculated for the test set. This result is better than the first result, 2.0% of errors for the test set, published for this data with the early version of the AdaBoost study by Schwenk and Bengio [15].

The UCI 'satellite' dataset has 4435 examples for training and 2000 for testing. The problem is to recognize 6 different classes of objects using satellite image data. Schwenk and Bengio results [9] were calculated with 36-30-15-6 MLPs compared to $36 \times 50 \times 6$ cascade-correlation architecture neural networks used in the current study. The efficient partition algorithm calculated 8.2% errors for the test set. The ASNN algorithm further improved this result to 7.8% test set errors, that is the best published result for this set.

Thus, ASNN method improved performance of MLPs for the real-world applications and calculated results comparable with the best published results for the well-known classification problems. Notice, that in the present study the calculated results were obtained using much simpler architecture of neural networks. The performance of the proposed method critically depends on the quality of neural network models. Thus it is possible that the ASNN applied to the ensemble of neural networks trained using AdaBoost algorithm would provide lower error rates for both UCI datasets.

4.3. SINE FUNCTION EXTRAPOLATION

An extrapolation problem was designed using two-dimensional sine function (Figure 2). The neural networks with two hidden neurons were trained using 50 data cases with $x = x_1 + x_2 < \pi/2$. The remaining 50 cases were not available for the model development. Both MLPs and ASNN provided a flat response, $y = 1$, for all values $x = x_1 + x_2 > \pi/2$ and a prediction of the extrapolated data was poor (rms = 0.54). However, if missed data were used as the memory of the proposed method, the network provided a satisfactory prediction of the function for $x = x_1 + x_2 > \pi/2$ values, rms = 0.034, without a need to retrain the neural network weights. Notice that this error coincided with the nearest-neighbor result calculated for this data in the one-dimensional space of x-values.

4.4 IDENTITY FUNCTION EXAMPLE

The neural networks trained using even numbers were unable to extrapolate odd numbers for the identity function example, thus replicating the results of Marcus [10]. In order to provide fresh information about the data to be extrapolated, the first

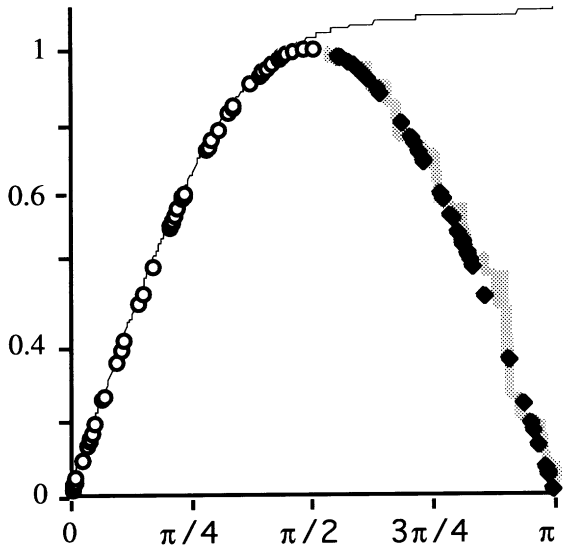

Figure 2. Sine function $y = \sin(x = x_1 + x_2)$ extrapolation. The MLPs (black line) trained with 50 cases (open circles) such as $x = x_1 + x_2 < \pi/2$ provided a poor extrapolation of the sine function for $x = x_1 + x_2 > \pi/2$ values. The response of ASNN with 50 cases (black rhombs) used as the memory of this method (without retraining neural network weights) is shown as the gray line.

number, 1, from the odd set [0 0 0 0 0 1] was used as the memory of ASNN. This dramatically changed prediction ability of networks for the test sets and all odd numbers were correctly mapped. This result was the same when any other odd number or a combination of such numbers were used as the memory.

4.5. GAUSS FUNCTION EXTRAPOLATION

In this example we considered a problem of prediction of a function that was changed (e.g. due to some non-stationarity in time) compared to the one used to develop neural network model. The neural networks with two hidden neurons were trained using two-dimensional sine function example (5) and then were used to predict 1000 data cases generated using

$$y = \exp(-(x_1 + x_2 - \pi/2)^2) \tag{6}$$

The neural networks provided large prediction error for such data, rms = 0.12. On the contrary, if 100 samples generated by (6), i.e. 'user's data', were added to the ASNN memory, this method predicted the test cases with rms = 0.03. Thus, without a need to retrain neural network weights it was possible to significantly improve performance of neural networks for a prediction of a function that was different from the one used to develop the neural networks model.

For this example one could calculate better result, e.g. rms = 0.015 for neural networks with two hidden neurons, by developing model from scratch, i.e. using 100

samples generated by Gauss function. However, in some cases the amount of available user's data could be insufficient to perform such modeling. For example, if only 10 cases generated by (6) were used to develop ASNN model from scratch, the neural networks calculated a poor result, rms = 0.34. This result could not be improved by a use of a different number (1 to 10) of hidden neurons. However, the approximation error of the ASNN developed with sine function data decreased in about two times, rms = 0.07, if the same 10 cases were used as the memory of these networks.

In the last three examples some properties of the test set (user's data) were different compared to those used in the training set. For example, some data were out of the range of input variables or target function was modified. However, other underlining properties (i.e., $x = x_1 + x_2$ or identity mapping of even numbers) were conserved for both training and test sets. Actual discrepancy between the user's data and the data used to develop the model was small. This made possible for ASNN to 'inherit' the knowledge about the conserved properties of the training set examples and to predict new data using the dissimilarities between experimental and calculated values of the user's examples. To this extent ASNN 'generalized' the previous experience obtained with the training sets and used it to provide a better model for the user's data.

4.6. PREDICTION OF LIPOPHILICITY OF CHEMICAL COMPOUNDS

ALOGPS program (http://www.vcclab.org/lab/alogps) for the calculation of the logarithm of n-octanol/water partition coefficient, $\log P$, and water solubility of molecules [16] was recently developed as described elsewhere [12, 17]. It was tested at BASF AG (Germany) with 6100 in-house compounds with known experimental $\log P$ values and the results of the analysis were reported to us [18]. The results calculated using the 'as is' version of the program were quite poor and only 49% of molecules were predicted with absolute error in the range from 0 to 0.5 log units. This is in agreement with our previous finding indicating a low predicting power of $\log P$ calculation methods whenever the training and testing set compounds are coming from the non-homologous series [12]. However, when the BASF in-house compounds were added to the memory of the ASNN (LIBRARY mode), the results estimated by the built-in leave-one-out method were improved considerably and 80% of molecules were predicted within the same error range. The chemical compounds usually remain a property of pharmaceutical companies and are not accessible for analysis and program development due to confidence restrictions. Thus, a use of ASNN provides an exceptional possibility to develop reliable methods for prediction of properties of such compounds without a need of their disclosure by companies.

5. Discussion

When predicting a property of an object, a person uses some global knowledge as well as known properties of other similar objects (local knowledge). The final predic-

tion represents some weighted sum of both these contributions, and (4) provides a simple model of such process. The underlining idea of this equation and of ASNN is that clustering in space of models is better than the clustering in space of input variables and that by considering the nearest neighbors in the space of models one can easily correct the bias of the final model. This equation also proposes how memoryless and memory-based methods could be combined to improve their performances.

ASNN has two phases in the learning process. The first phase includes training of neural network ensemble to correctly represent topology of the space. This is a difficult task and it has a long training time, e.g. it can correspond to acquisition of driving experience by a person. The second phase provides bias correction according to (4). This phase is used to optimize the number of neighbors, k, in order to fit better the local features of the analyzed data. There are some clear situations in which user should take into account local properties of the model to achieve an optimal performance but in which the retraining of the global model is impossible due to limitations in time and resources or due to a limited number of available samples.

For example, consider a situation when a small car driver rents a large van. Even though that the new automobile has some specific features, only a slight adaptation of local rules, corresponding to local bias correction, is required for him to drive the van. Another similar example corresponds to the situation when a UK driver (left-side driving) comes to France (right-side driving), i.e. when he has to discard his left-side driving experience. The including of the local corrections makes it possible to adapt the 'old' left-side driving experience with new knowledge and to achieve quickly a good performance in the changed environment. In the next example the same driver has to cross a river on foot. Again, the basic dynamic of his movement that involves very complex interaction of muscles in the body remains the same as for walking on the ground. However, some important changes to this behavior will be done to move in water. Notice, that a requirement to re-train the 'walking' network in order to optimize driver's performance for movement in water does not make sense, since the crossing of rivers is a very seldom event in his live. Therefore, a simple correction of his movement would be sufficient. On the other hand, if the driver starts living in France permanently, the local features particular to driving in this new environment will be eventually incorporated into the global model.

It is quite difficult for these three examples to separate problem of extrapolation of new data and problem of prediction of function that is changed (in time) compared to the function used to develop a model. The introduced neural network is able to cope efficiently with both these problems. It is also important to mentions that using such king of 'on-fly' learning, the ASNN is no more restricted to the range of values used in the training set.

The ASNN has solid neurophysiological background. The recent theories of brain coding suggest an importance of temporal coding for information processing in brain [19, 20]. Analysis of speed of processing in the human visual system and particular features of this system (e.g., ability to recognize the same signals at different

brightness of images) suggests an importance of the rank coding for information processing [21]. To this extent, each network in the proposed method corresponds to one spiking neuron and the ASNN itself corresponds to an ensemble of spiking neurons processing some specific task. Thus assumption that the rank of spiking neuron corresponds to its temporal delay (the first firing neuron has the highest rank) makes the ASNN approach similar to the rank coding model of Thorpe [22]. Therefore a detection of nearest neighbors by ASNN could be considered as recognition of associative patterns performed by the brain. The proposed model does not explain how such patterns are stored in the brain but suggests a mechanism of how these patterns can be used to improve regression or classification with neural networks.

It will also be interesting to see, if other non-linear global approximation methods, such as learning vector quantization [23], support vector machine [24], polynomial neural networks [25], etc., could be used instead of the feed-forward neural networks. Another possibility to further develop the idea of ASNN is to investigate whether other local regression techniques, e.g. multiple linear regression analysis or even the ASNN itself, could be used instead of the KNN. The initial input parameters could be also provided in addition to or instead of the correlation coefficients.

The proposed method does not require more computational resources compared to the traditional application of ensemble of neural networks. The KNN clustering is computationally less expensive (orders of magnitude) than the training of the ensemble. In all reported studies a comparison of ASNN was done with the corresponding predictions of a similar size ensemble of neural networks, e.g., Schwenk and Bengio [13] also calculated their results using an ensemble of 100 neural networks. A more detailed case study of the ASNN parameters, such as a use of different similarity measures, number of neural networks in ensemble and strategies to select the number of nearest neighbors can be found elsewhere [17].

Conclusions

The ASNN improves prediction of traditional neural network by correcting for bias of this method. This can provide a significant increase in the performance of the neural networks, as demonstrated using sine function example and UCI benchmark datasets. If some new data are available and if their properties are changes compared to the training set, the proposed method provides a fast adaptation of the previously developed neural networks to such data without a need to retrain their weights.

Acknowledgements

This study was partially supported by Virtual Computational Chemistry Laboratory project, INTAS-INFO 00-0363, and SNSF 7-IP-062620 grants. I thank Igor V. Litvinyuk, Steacie Institute for Molecular Sciences, Canada, and Roman M. Borisyuk, University of Plymouth, UK, for their helpful suggestions, Hugo

Kubinyi (BASF, Germany) for the testing of the ALOGPS program and anonymous reviewers for their valuable comments.

References

1. Dasarthy, B.: *Nearest neighbor (NN) norms*, IEEE Computer Society Press, Washington, DC, 1991.
2. Härdle, W.: *Smoothing techniques with implementation in S*, Springer-Verlag, New York, 1990.
3. Lawrence, S., Tsoi, A. C. and Back, A. D.: Function approximation with neural networks and local methods: bias, variance and smoothness, In: P. Bartlett, A. Burkitt and R. Williamson (eds), *Australian Conference on Neural Networks*, Australian National University, Australian National University, 1996, pp. 16–21.
4. Geman, S., Bienenstock, E. and Doursat, R.: Neural networks and the bias/variance dilemma, *Neural Computation* **4** (1992), 1–58.
5. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: *Numerical Recipes in C*, Cambridge University Press, New York, 1994.
6. Tetko, I. V., Livingstone, D. J. and Luik, A. I.: Neural network studies. 1. Comparison of overfitting and overtraining, *Journal of Chemical Information & Computer Sciences* **35** (1995), 826–833.
7. Bishop, M.: *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.
8. Tetko, I. V. and Villa, A. E. P.: Efficient partition of learning data sets for neural network training, *Neural Networks* **10** (1997), 1361–1374.
9. Schwenk, H. and Bengio, Y.: Boosting neural networks, *Neural Computation* **12** (2000), 1869–1887.
10. Marcus, G. F.: Rethinking eliminates connectionism, *Cognitive Psychology* **37** (1998), 243–282.
11. Blake, E. K. and Merz, C. UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
12. Tetko, I. V., Tanchuk, V. Y. and Villa, A. E. P.: Prediction of n-octanol/water partition coefficients from physprop database using artificial neural networks and E-state indices, *J. Chem. Inf. Comput. Sci.* **41** (2001), 1407–1421.
13. Tetko, I. V. and Villa, A. E. P.: An efficient partition of training data set improves speed and accuracy of cascade-correlation algorithm, *Neural Processing Letters* **6** (1997), 51–59.
14. Fahlman, S. and Lebiere, C.: The cascade-correlation learning architecture, *NIPS* **2** (1990), 524–532.
15. Schwenk, H. and Bengio, Y. Adaptive boosting of neural networks for character recognition, Université de Montréal, Montréal, 1997, pp. 1–9.
16. Tetko, I. V., Tanchuk, V. Y., Kasheva, T. N. and Villa, A. E.: Internet software for the calculation of the lipophilicity and aqueous solubility of chemical compounds, *Journal of Chemical Information & Computer Sciences* **41** (2001), 246–252.
17. Tetko, I. V. and Tanchuk, V. Y.: Application of associative neural networks for prediction of lipophilicity in ALOGPS 2.1 program, *Journal of Chemical Information & Computer Sciences* in press (2002).
18. Tetko, I. V.: Neural network studies. 4. Introduction to associative neural networks, *Journal of Chemical Information & Computer Sciences* **42** (2002), 717–728.
19. Abeles, M.: *Corticotronics: Neural circuits of the cerebral cortex*, Cambridge University Press, New York, 1991.

20. Villa, A. E. P., Tetko, I. V., Hyland, B. and Najem, A.: Spatiotemporal activity patterns of rat cortical neurons predict responses in a conditioned task, *Proceedings of the National Academy of Sciences of the Unites States of America* **96** (1999), 1106–1111.
21. Thorpe, S., Fize, D. and Marlot, C.: Speed of processing in the human visual system, *Nature* **381** (1996), 520–522.
22. Gautrais, J. and Thorpe, S.: Rate coding versus temporal order coding: a theoretical approach, *Biosystems* **48** (1998), 57–65.
23. Kohonen, T.: *Self-Organizing Maps*, Springer, Berlin, 2001.
24. Vapnik, V.: *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
25. Tetko, I. V., Aksenova, T. I., Volkovich, V. V., Kasheva, T. N., Filipov, D. V., Welsh, W. J., Livingstone, D. J. and Villa, A. E. P.: Polynomial neural network for linear and non-linear model selection in quantitative-structure activity relationship studies on the Internet, *SAR & QSAR in Environmental Research* **11** (2000), 263–280.
26. Breiman, L.: Arcing classifiers, *Annals of Statistics* **26** (1998), 801–824.
27. Freund, Y. and Schapire, R. E.: Experiments with a new boosting algorithm, In: L. Saitta (ed.), *Machine Learning: Proceedings of the Thirteen National Conference*, Morgan Kaufmann, 1996, pp. 148–156.