Evolving Connectionist Systems: A Theory and a Case Study on Adaptive Speech Recognition

Nikola Kasabov
Department of Information Science
University of Otago, P.O Box 56, Dunedin, New Zealand
Phone: +64 3 479 8319, fax: +64 3 479 8311
nkasabov@otago.ac.nz

Abstract.

The paper introduces evolving connectionist systems (ECOS) as an effective approach to building on-line, adaptive intelligent systems. ECOS evolve through incremental, hybrid (supervised/unsupervised), on-line learning. They can accommodate new input data, including new features, new classes, etc. through local element tuning. New connections and new neurons are created during the operation of the system. The ECOS framework is presented and illustrated on a particular type of evolving neural networks - evolving fuzzy neural networks (EFuNNs). EFuNNs can learn spatial-temporal sequences in an adaptive way, through one pass learning. Rules can be inserted and extracted at any time of the system operation. The characteristics of ECOS and EFuNNs are illustrated on a case study of adaptive, phoneme-based spoken language recognition.

1. Introduction

The complexity and dynamics of real-world problems, especially in engineering and manufacturing, require sophisticated methods and tools for building on-line, adaptive intelligent systems (IS). Such systems should be able to grow as they operate, to update their knowledge and refine the model through interaction with the environment. This is especially crucial when solving AI problems such as adaptive speech and image recognition, multi-modal information processing, adaptive prediction, adaptive on-line control, intelligent agents on the WWW. Seven major requirements of the present IS (that are addressed in the ECOS framework presented later) are listed below [20-22]:

(1) IS should *learn fast* from a large amount of data (using fast training, e.g. one-pass training).
(2) IS should be able to *adapt incrementally* in both real time, and in an on-line mode, where new data is accommodated as they become available. The system should tolerate and accommodate imprecise and uncertain facts or knowledge and refine its knowledge.
(3) IS should have an *open structure* where new features (relevant to the task) can be introduced at a later stage of the system's operation. IS should dynamically create new modules, new inputs and outputs, new connections and nodes. That should occur either in a supervised, or in an unsupervised mode, using one modality or another, accommodating data, heuristic rules, text, images, etc.
(4) IS should be *memory-based*, i.e. they should keep a reasonable track of information that has been used in the past and be able to retrieve some of it for the purpose of inner refinement, or for answering an external query.
(5) IS should improve continuously (possibly in a life–long mode) through active *interaction* with other IS and with the environment they operate in.
(6) IS should be able to *analyse themselves* in terms of behaviour, global error and success; to explain what has been learned; to make decisions about its own improvement; to manifest introspection.
(7) IS should adequately represent *space and time* in their different scales; should have parameters to represent such concepts as spatial distance, short-term and long-term memory, age, forgetting, etc.

Several investigations proved that the most popular neural network models and algorithms are not suitable for adaptive, on-line learning [30,17,11]. At same time some of the seven issues above have been acknowledged and addressed in the development of several NN models for adaptive learning and for structure and knowledge manipulation, for example: learning through self-organisation [4,23,24,10,], incremental learning [4,5], on-line learning [11,15,9,31], local learning [3], learning in growing structures [10,8], learning with pruning [25,16,13,29], evolutionary learning [12,7,6], knowledge-based NN [2,18,17], neuro-fuzzy systems [34,26,17-19]. A framework called ECOS (Evolving COnnectionist Systems) that addresses all seven issues above is introduced in the paper and illustrated on the case study problem of adaptive phoneme-based speech recognition.

2. The ECOS framework

Evolving connectionist systems (ECOS) are systems that evolve in time through interaction with the environment [20-22]. They have some (genetically) pre-defined parameters (knowledge) but they also learn and adapt as they operate. In contrast with the evolutionary systems they do not necessarily create copies of individuals and select the best ones for the future. They emerge, evolve, develop, unfold through both innateness and learning, and through changing their structure in order to better represent data [20-22]. ECOS learn in an on-line, and in a knowledge–based mode, so they can accommodate any new incoming data from a data stream, and the learning process can be expressed as a process of rule manipulation. ECOS are multi-level, multi-modular structures where many neural network modules (NNM) are connected with inter-, and intra- connections. ECOS do not have a clear multi-layer structure, but rather a modular, "open" structure. Initially an ECOS has a pre-defined structure of some NNMs, each of them being a mesh of nodes (neurons) and very few connections defined through prior knowledge, or "genetic" information. Gradually, the system becomes more and more "wired" through self-organisation, and through creation of new NNM and new connections.

The ECOS functioning is based on the following *general principles*:
(1) ECOS evolve incrementally in an on-line, hybrid, adaptive *supervised/unsupervised mode* through accommodating more and more examples when they become known from a continuous input data stream. .
(2) ECOS are memory-based and store exemplars (prototypes, rules) that represent groups of data from the data stream. New input vectors are stored in the NNMs based on their similarity to previously stored data both on the input and the desired output information. A node in an NNM is created and designated to represent an individual example if it is significantly different from the previously used examples (with a level of differentiation set through dynamic parameters). Learning is based on *locally tuned* elements from the ECOS structure thus making the learning process fast for real-time parallel implementation.
(3) There are *three levels* at which ECOS are functionally and structurally defined:
(a) Parameter (gene) level, i.e. a chromosome contains genes that represent certain parameters of the whole systems, such as: type of the structure (connections) that will be evolved; learning rate; forgetting rate; size of a NNM; NNM specialisation, thresholds that define similarity; error rate that is tolerated, and many more. The values of the genes are relatively stable, but can be changed through genetic operations, such as mutation of a gene, deletion and insertion of genes that are triggered by the self analysis module as a result of the overall performance of the ECOS.
(b) Representation (synaptic) level, that is the information contained in the connections of the NNM. This is the long-term memory of the system where exemplars of data are stored. They can be either retrieved to answer an external query, or can be used for internal ECOS refinement.
(c) Behavioural (neuronal activation) level, that is the short-term activation patterns triggered by input stimuli. This level defines how well the system is functioning in the end.
(4) ECOS evolve through *learning (growing), forgetting (pruning), and aggregation*, that are both defined at a genetic level and adapted during the learning process. ECOS allow for: creating/connecting neurons; removing neurons and their corresponding connections that are not actively involved in the functioning of the system thus making space for new input patterns to be learned; aggregating nodes into bigger-cluster nodes.
(5) There are two global modes of learning in ECOS:
(a) Active learning - learning is performed when a stimulus (input pattern) is presented and kept active.
(b) Passive (inner, ECO) learning mode - learning is performed when there is no input pattern presented to the ECOS. In this case the process of further elaboration of the connections in ECOS is done in a passive learning phase, when existing connections, that store previously fed input patterns, are used as "echo" (here denoted as ECO) to reiterate the learning process (see for example fig.9 explained later).
There are two types of ECO training:

- *cascade eco*-training: a new connectionist structure (a NN) is created in an on-line mode when conceptually new data (e.g., a new class data) is presented. The NN is trained on the positive examples of this class, on the negative examples from the following incoming data, and on the negative examples from previously stored patterns in previously created modules.
- *'sleep' eco*-training: NNs are created with the use of only partial information from the input stream (e.g., positive class examples only). Then the NNs are trained and refined on the stored patterns (exemplars) in other NNs and NNMs (e.g., as negative class examples).

(6) ECOS provide explanation information extracted from the NNMs through the self-analysis/ rule extraction module. Generally speaking, ECOS learn and store knowledge, rules, rather than individual examples or meaningless numbers.
(7) The ECOS principles above are based on some biological facts and biological principles (see for example [2,33,27]).

Implementing the ECOS framework and the NNM from it requires connectionist models that comply with the ECOS principles. One of them, called evolving fuzzy neural network (EFuNN), is presented in the next section.

3. Evolving Fuzzy Neural Networks EFuNNs

3.1. General principles of EFuNNs

Fuzzy neural networks are connectionist structures that implement fuzzy rules and fuzzy inference [26,34,17]. FuNNs represent a class of them [17,19]. EFuNNs are FuNNs that evolve according to the ECOS principles. EFuNNs were introduced in [20-22] where preliminary results were given. Here EFuNNs are further developed and illustrated on adaptive speech recognition tasks. EFuNNs have a five-layer structure, similar to the structure of FuNNs (fig.1). But here nodes and connections are created/connected as data examples are presented.

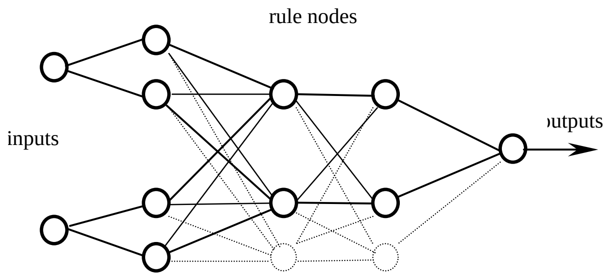

Fig.1 The five layer basic structure of the EFuNN.

An optional short-term memory layer can be used through a feedback connection from the rule (also called, case) node layer. The layer of feedback connections can be used if temporal relationships between input data are to be memorised structurally as it is the case in speech recognition.

The input layer represents input variables. The second layer of nodes (fuzzy input neurons, or fuzzy inputs) represents fuzzy quantization of each input variable space. For example, two fuzzy input neurons can be used to represent "small" and "large" fuzzy values. Different membership functions (MF) can be attached to these neurons (triangular, Gaussian, etc.). The number and the type of MF can be dynamically modified in an EFuNN. The task of the fuzzy input nodes is to transfer the input values into membership degrees to which they belong to the MF. The third layer contains rule (case) nodes that evolve through supervised/unsupervised learning. The rule nodes represent prototypes (exemplars, clusters) of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and fuzzy output spaces (see fig.2). Each rule node r is defined by two vectors of connection weights – W1(r) and W2(r), the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure within a local area of the problem space. The fourth layer of neurons represents fuzzy quantization for the output variables, similar to the input fuzzy neurons representation. The fifth layer represents the real values for the output variables.

Each rule node, e.g. r_j, represents an association between a hyper-sphere from the fuzzy input space and a hyper-sphere from the fuzzy output space (fig.2), the W1(r_j) connection weights representing the co-ordinates of the center of the sphere in the fuzzy input space, and the W2 (r_j) – the co-ordinates in the fuzzy output space. The radius of an input hyper-sphere of a rule node is defined as (1-Sthr), where Sthr is the sensitivity threshold parameter defining the minimum activation of a rule node (e.g., r1, previously evolved to represent a data point (Xd1,Yd1)) to an input vector (e.g., (Xd2,Yd2)) in order for the new input vector to be associated with this rule node. For example, two pairs of fuzzy input-output data vectors **d1**=(Xd1,Yd1) and **d2**=(Xd2,Yd2) will be allocated to the first rule node r_1 if they fall into the r_1 input sphere and in the r_1 output sphere, i.e. the local normalised fuzzy difference between Xd1 and Xd2 is smaller than the radius *r* and the local normalised fuzzy difference between Yd1 and Yd2 is smaller than an error threshold Errthr. The local normalised fuzzy difference between two fuzzy membership vectors **d1f** and **d2f** that represent the membership degrees to which two real values d1 and d2 data belong to the pre-defined MF, are calculated as D(**d1f**,**d2f**) = sum(abs(**d1f** - **d2f**))/sum(**d1f** + **d2f**)). For example, if **d1f**=(0,0,1,0,0,0) and **d2f**=(0,1,0,0,0,0) (see fig.3), than D(d1,d2) = (1+1)/2=1 which is the maximum value for the local normalised fuzzy difference. If data example **d1** = (Xd1,Yd1), where Xd1 and Xd2 are correspondingly the input and the output fuzzy membership degree vectors, and the data example is associated with a rule node r_1 with a centre r_1^1, than a new data point **d2**=(Xd2,Yd2) will be associated with this rule node too. Through the process of associating (learning) of new data points to a rule node, the centres of this node hyper-spheres adjust in the fuzzy input space depending on a learning rate lrn1, and in the fuzzy output space depending on a learning rate lr2, as it is shown in fig.4a on the two data points **d1** and **d2**. The adjustment of the centre r_1^1 to its new position r_1^2 can be represented mathematically by the change in the connection weights of the rule node r_1 from W1(r_1^1) and W2(r_1^1) to W1(r_1^2) and W2(r_1^2) according to the following vector operations:

$$W2(r_1^2) = W2(r_1^1) + lr2.\ Err(Yd1,Yd2).\ A1(r_1^1)$$
$$W1(r_1^2)=W1(r_1^1) + lr1.\ Ds(Xd1,Xd2),$$

where: Err(Yd1,Yd2)= Ds(Yd1,Yd2)=Yd1-Yd2 is the signed value rather than the absolute value of the fuzzy difference vector; A1(r_1^1) is the activation of the rule node r_1^1 for the input vector Xd2.

While the connection weights from W1 and W2 capture spatial characteristics of the learned data (centres of hyper-spheres), the temporal layer of connection weights (e.g. W3) captures temporal dependencies between consecutive data examples. If the winning rule node at the moment (t-1) (to which the input data vector at the moment (t-1) was associated) was r1=inda1(t-1), and the winning node at the moment t is r2=inda1(t), then a link between the two nodes is established as follows:

$W3(r1,r2)^{(t)} = W3(r1,r2)^{(t-1)} + lr3. A1(r1)^{(t-1)} A1(r2))^{(t)}$,

where: $A1(r)^{(t)}$ denotes the activation of a rule node r at a time moment (t); lr3 defines the degree to which the EFuNN associates links between rules (clusters, prototypes) that include consecutive data examples (if lr3=0, no temporal associations are learned in an EFuNN structure).

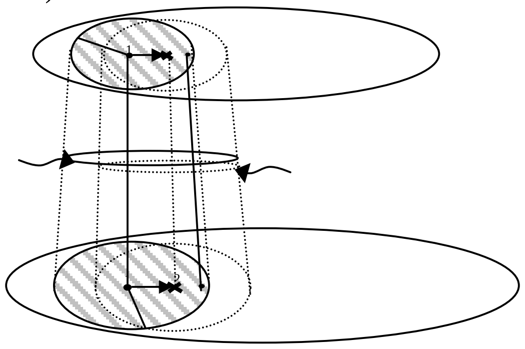

Fig.2. Input /output mapping in an EFuNN and learning through adjustment of the centres of the spheres represented by the connection weights of the rule nodes

The EFuNN algorithm, to evolve EFuNNs from incoming examples, is presented in [20-22]. Several simulators implementing it are available as part of the New Zealand Repostory of Intelligent Connectionist-based Information Systems (RICBIS) from:
http://divcom.otago.ac.nz/infosci/CBIIS.html

There are different ways to locate rule nodes in an EFuNN rule node space: linear – the nodes are located ton represent exemplars in the order of their presentation; spatial – a new node is located nearest to the highest activation node for the current input vector.

3.2. Learning modes in EFuNN. Rule insertion, rule extraction and aggregation.

Different learning, adaptation and optimisation strategies and algorithms can be applied on an EFuNN structure for the purpose of its evolving. These include:

- *Active learning* , e.g. the EFuNN algorithm;
- *Passive learning* (i.e., *cascade-eco*, and *sleep-eco* learning) [20-22]
- *Rule insertion into EFuNNs, Rule extraction and aggregation:* Each rule node *r*, which represents a prototype, rule, exemplar from the problem space, can be described by its connection weights W1(r) and W2 (r) that define the association of the two corresponding hyper-spheres from the fuzzy input and the fuzzy output problem spaces. The association is expressed as a fuzzy rule.
- *Aggregation and abstraction through ECO-learning*: Aggregation of rule nodes to represent association of larger hyper-spheres from the input and the output space can be achieved through the use of the ECO learning method, when the connection weights $W1^{(1)}$ and $W2^{(1)}$ of an evolved EFuNN1 are used as fuzzy exemplars to evolve an EFuNN2 for smaller values of the sensitivity threshold Sthr and the error threshold Errthr (see fig.9). This process can be continued further to evolve a new EFuNN3 with smaller number of rule nodes, therefore smaller number of rules, and so on.
- *Extracting rules for learning temporal pattern correlation:*
  Through analysis of the weights W3 of an evolved EFuNN, temporal correlation between time consecutive exemplars can be expressed in terms of rules and conditional probabilities, e.g.:
  IF $(W1(r1),W2(r1))^{(t-1)}$ THEN $(W1(r1),W2(r2))^{(t)}$ (0.3)

The meaning of the above rule is that examples that belong to the rule (prototype) r1 follow in time examples from the rule prototype r2 with a relative conditional probability of 0.3.

- *Changing MF during operation* [22].
- *On-line parameter optimisation*. Once set, the values for the EFuNN parameters will need to be optimised during the learning process. Optimisation can be done through analysis of the behaviour of the system and through a feedback connection from the higher level modules. Genetic algorithms (GA) can also be applied to optimise the EFuNNs structural and functional parameters based on either standard GA algorithms, or on their possible modifications for dynamic, on-line application.
- *Learning and pruning:* With the learning and pruning operations as part of the EFuNN learning algorithm, and with some additional adaptation techniques, an EFuNN can dynamically organise its structure to learn from data in an adaptive, continuous, incremental, life-long learning mode.

3.3. EFuNNs as universal learning machines. Local and global generalisation

EFuNNs are designed to work in an on-line mode, with a continuous input data stream. An EFuNN is trained (evolved) on input-output vectors of data available over time. Then it is used to generalise on new incoming data Xd for which the output is not known. Once the output vector Yd for the new input data becomes known, the input-output pair (Xd,Yd) is accommodated in the EFuNN structure, which is then used on the next input data, and so on. In an on-line learning of an EFuNN local generalisation is calculated, i.e., the generalisation over the next incoming data. Global generalisation, i.e. the generalisation over a whole set of data (test data) can also be evaluated. The generalisation ability of EFuNNs depends on the learning and pruning coefficients which can be dynamically adjusted in an ECOS architecture through a feedback connection from the higher level decision module or through optimisation techniques. EFuNNs are universal learning machines that can learn, subject to a chosen degree of accuracy, any data set, regardless of the class of problems (function approximation, time series prediction, classification, etc.).

4. Evolving Systems for On-line Incremental Learning for Adaptive Speech Recognition

Adaptive speech recognition is concerned with the development of speech recognition systems that: (1) can adapt to new pronunciation (of the same, or a new speaker); (2) can enlarge their vocabulary of words in an on-line mode; (3) can acquire new languages. Here, EFuNNs are illustrated on the problem of phoneme adaptation.

It is well known that, there are a lot of variations in the pronunciation of phones of the same phonemes, and at the same time there are similarities in the pronunciation of phones of different phonemes. These make the recognition of phonemes a very difficult task. Four phoneme data is used here (the same four phonemes as in the example from section 3, but here taken from the words 'pit', 'pet', 'pat' and 'bean' from the same data base, same two speakers [32]). Figures 3 illustrate the temporal variability of the /I/ phoneme data (new speaker, not in the database, pronouncing the word 'sit') and the /e/ phoneme data (from the word 'get', speaker 17 from the database) within a small time interval. Fig.3 shows the values of the 26 mel-scale coefficients of the phoneme /I/ data for each of ten consecutive time frames (each of them 11.6 msec long). It can be seen that while there is similarity in the mel-scale vector patterns, there is a significant difference in the values of the main mel coefficients.

In the experiment below the data is grouped into two data sets – a set A, that constitutes a first pronunciation of the four phonemes, and a set B – a second pronunciation of the same words by the same speakers. Once evolved on set A the system will be tested on set B and if it does not perform well it will be adapted to set B. The level of forgetting on the set A will be tested. The following numbers of 78–element frame vectors are used as positive examples (and negative examples in brackets): /I/ - 174 (85); /e/ - 253 (124); /ae/ - 285(138); /i/ - 325 (159). The data is taken from the Otago Speech Corpus:
http://divcom.otago.ac.nz:800/com/infosci/kel/CBIIS.html
Initially four EFuNNs were evolved from the set A through one pass of training for the following parameter values: linear activation functions; SThr=0.5; lr1=lr2=0.5; lr3=0; no pruning; Errthr=0.01. The classification rate was evaluated on set A (to evaluate the training error), and on set B (to evaluate the generalisation of the EFuNNs over a new articulation data of the same speakers (tabl.1). Then all EFuNNs were further trained for one pass on the set B to adapt to the new articulation data. After the additional training the EFuNNs were tested again on set A and set B. The classification rate significantly improved on both set A and set B. This experiment shows that EFuNNs can successfully adapt to new pronuncition without forgetting previous ones. When temporal links were evolved, for a small learning rate of lr3=0.01, the classification accuracy further improved which was expected after having seen the temporal variations within the phones of same phonemes from fig.3a,b (tabl.2).

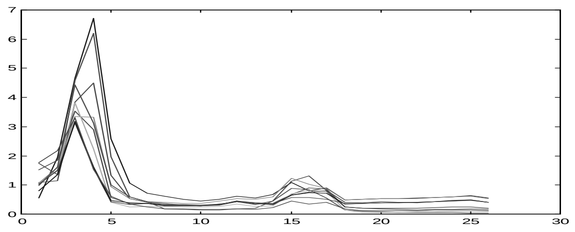

Fig. 3 Phoneme /I/ from 'sit'- 10 consecutive mel scale vectors, each of 26 elements;

Table 1

	4 EFuNNs are evolved for one pass on A and tested on A and on B (in%)		The evolved on A EFuNNs are adjusted for one pass on B and tested on A and B	
	on A	on B	on A	on B
I	95(99)	71(99)	96(99)	97(99)
e	95(97)	74(96)	97(98)	91(98)
ae	98(99)	81(93)	99(99)	94(98)
I	93(95)	76(82)	92(98)	94(95)

Table 2.

	Temporal EFuNNs are evolved for one pass on A and tested on A and on B (lr3=0.001)		The temporal EFuNNs that were evolved on A are adjusted for one pass on B and tested on A and B (lr3=0.001)	
	on A	on B	on A	on B
I	94(99)	74(99)	96(99)	98(99)
e	95(97)	80(96)	97(98)	93(98)
ae	97(99)	81(94)	99(99)	93(98)
i	95(96)	75(82)	94(98)	96(95)

EFuNNs are currently being used for building a general framework of an adaptive phoneme-based speech recognition system that adapts its phoneme modules after every unsuccessful recognition attempt.

5. Conclusions

ECOS and EFuNNs are suitable models for on-line, adaptive, learning, knowledge-based systems. Further development in this area includes building ECOS for multi-modal, multilingual spoken language processing systems.

Acknowledgements

This research is part of a research programme funded by the New Zealand Foundation for Research Science and Technology, contract UOO808.

References

1. Amari, S. and Kasabov, N. eds, "Brain-like Computing and Intelligent Information Systems", Springer Verlag,1998.
2. Arbib, M. (ed) The Handbook of Brain Theory and Neural Networks,The MIT Press, 1995.
3. Bottu and Vapnik, "Local learning computation", Neural Computation, 4, 888-900 (1992)
4. Carpenter, G. and Grossberg S., Pattern recognition by self-organizing neural networks , The MIT Press, Cambridge, Massachusetts (1991)
5. Carpenter, G. S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, "FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps," IEEE Transactions of Neural Networks , vol.3, No.5, 698-713 (1991).
6. DeGaris, H., "Circuits of Production Rule - GenNets – The genetic programming of nervous systems", in: Albrecht, R., Reeves, C. and Steele, N. (eds) Artificial Neural Networks and Genetic Algorithms, Springer Verlag (1993)
7. Edelman, G., Neuronal Darwinism: The theory of neuronal group selection, Basic Books (1992).
8. Fahlman, C., and C. Lebiere, "The Cascade- Correlation Learning Architecture", in: Turetzky, D (ed) Advances in Neural Information Processing Systems, vol.2, Morgan Kaufmann, 524-532 (1990).
9. Freeman, J., D. Saad, "On-line learning in radial basis function networks", Neural Computation vol. 9, No.7 (1997).
10. Fritzke, B. "A growing neural gas network learns topologies", Advances in Neural Information Processing Systems, vol.7 (1995).
11. Gaussier, T., and S. Zrehen, "A topological neural map for on-line learning: Emergence of obstacle avoidance in a mobile robot", In: From Animals to Animats No.3, 282-290, (1994).
12. Goldberg, D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley (1989)
13. Hassibi and Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," in: Advances in Neural Information Processing Systems, 4, 164-171, (1992).
14. Hech-Nielsen, R. "Counter-propagation networks", IEEE First int. conference on neural networks, San Diego, vol.2, pp.19-31 (1987)
15. Heskes, T.M., B. Kappen, "On-line learning processes in artificial neural networks", in: Math. foundations of neural networks, Elsevier, Amsterdam, 199-233, (1993).
16. Ishikawa, M., "Structural Learning with Forgetting", Neural Networks 9, 501-521, (1996).
17. Kasabov, N., Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA, (1996).
18. Kasabov, N. "Adaptable connectionist production systems". Neurocomputing, 13 (2-4) 95-117, (1996).
19. Kasabov, N., "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems 82 (2) 2-20 (1996).
20. Kasabov, N. The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems, Journal of Advanced Computational Intelligence, 2 (6) 1998, 1-8
21. Kasabov, N., "ECOS: A framework for evolving connectionist systems and the ECO learning paradigm", Proc. of ICONIP'98, Kitakyushu, Japan, Oct. 1998, IOS Press, 1222-1235
22. Kasabov, N., "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation", in: in: Yamakawa and Matsumoto (eds), Methodologies for the Conception, design and Application of Soft Computing, World Scientific, 1998, 271-274
23. Kohonen, T., "The Self-Organizing Map", Proceedings of the IEEE, vol.78, N-9, pp.1464-1497, (1990).
24. Kohonen, T., Self-Organizing Maps, second edition, Springer Verlag, 1997
25. Le Cun, Y., J.S. Denker and S.A. Solla, "Optimal Brain Damage", in: Touretzky, D.S., ed., Advances in Neural Information Processing Systems, Morgan Kaufmann, 2, 598-605 (1990).
26. Lin, C.T. and C.S. G. Lee, Neuro Fuzzy Systems, Prentice Hall (1996).
27. Quartz, S.R., and T.J. Sejnowski, "The neural basis of cognitive development: a constructivist manifesto", Behavioral and Brain Science, to appear.
28. R. Jang, "ANFIS: adaptive network-based fuzzy inference system", IEEE Trans. on Syst., Man, Cybernetics, 23(3), May-June, 665-685, (1993).
29. Reed, R., "Pruning algorithms - a survey", IEEE Trans. Neural Networks, 4 (5) 740-747, (1993).
30. Robins, A. and Frean, M. "Local learning algorithms for sequential learning tasks in neural networks, Journal of Advanced Computational Intelligence, vol.2, 6 (1998)
31. Saad, D. (ed) On-line learning in neural networks, Cambridge University Press, 1999
32. Sinclair, S., and C. Watson, "The Development of the Otago Speech Database", In Kasabov, N. and Coghill, G. (Eds.), Proceedings of ANNES '95, Los Alamitos, CA, IEEE Computer Society Press (1995).
33. Wong, R.O. "Use, disuse, and growth of the brain", Proc. Nat. Acad. Sci. USA, 92 (6) 1797-99, (1995).
34. Yamakawa, T., H. Kusanagi, E. Uchino and T. Miki, "A new Effective Algorithm for Neo Fuzzy Neuron Model", in: Proceedings of Fifth IFSA World Congress, 1017-1020, (1993).