Modified Cascade Correlation Learning Architecture using Non-monotonic Activation Function

[1] Sang-Wha Lee [2] Dong-Yoon Kim [2] Min-Koo Kim [1] Claudio Moraga

[1] Department of Computer Science I
University of Dortmund
44221 Dortmund. Germany
E-mail: {lee|moraga}@ls1.
informatik.uni-dortmund.de

[2] Department of Electrical
and Computer Engineering
University of Ajou
Suwon, Korea
E-mail: {dykim|minkoo}@ce.ajou.ac.kr

Abstract

This paper presents a modified Cascade Correlation learning architecture using new activation functions, which herein will be called CosExp- and CosGauss functions. The modified Cascade Correlation is a supervised learning algorithm that automatically determines the size and topology of the network. The modified Cascade Correlation adds new hidden units one by one. Whenever a hidden unit has to be added, the modified Cascade Correlation learning algorithm automatically determines whether the network topology grows vertically or horizontally, and whether the added hidden unit should be feedforward or recurrent. The modified Cascade Correlation leads to a compact and elegant network. Experimental results on the balance scale benchmark problem are provided.

1 Modified Cascade Correlation Learning Architecture

We present a strategy for minimizing the number of hidden layers and hidden units required by the Cascade Correlation learning algorithm [1] [2] [3]. Readers are expected to be familiar with this type of neural network. To modify the Cascade Correlation learning architecture a pool of candidate units is divided into four groups. The candidate units of the first and third groups are connected with all input units, the bias units and all of the previously added hidden units. By the second and fourth groups, the candidate units receive connections from the input units, the bias units and from all previously added hidden units, that are not in the same hidden layer. Whenever a hidden unit has to be added, the modified Cascade Correlation automatically determines whether the network topology grows vertically or horizontally, and whether the added hidden unit should be feedforward or recurrent. In each group, the number of the candidate units must be the same e.g. four to four or eight to eight. Figure 1 shows the arrangement of four candidate units of each group in a pool, in which *v* denotes vertical and *h* represents horizontal. The candidate unit with the arrow is recurrent.

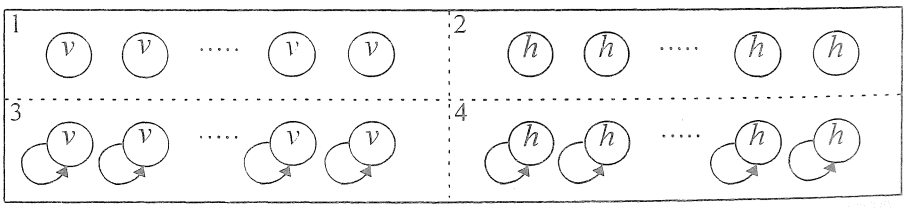


Fig. 1: Arrangement of the candidate units in four groups of the pool

All candidate units of the same group receive the same input signals and have the same residual error for each training pattern. Since they do not interact with one another or affect the active network during training, all of these candidate units of the pool can be trained in parallel; whenever we decide that no further progress is being made, we install the candidate whose correlation score with respect to the residual error is the best. Whenever a hidden unit is added, the modified Cascade Correlation network is developed as follows: the network grows vertically with a feedforward unit if the candidate unit is chosen from the first group of the pool, and with a recurrent one if the candidate unit is chosen from the third group of the pool. The network is generated horizontally with a feedforward unit if the candidate unit comes from the second group of the pool, and with a recurrent one if the candidate unit comes from the fourth group of the pool.

To develope the network, a candidate unit is chosen from the first or the third groups of the pool. This enables a hidden layer to be generated. Figure 2 shows the initial state of a modified Cascade Correlation network. Figure 3 (a) and (b) represent the adding of the first hidden unit without recurrent and with recurrent, respectively. To continue generating the network, candidate units from all groups are chosen. Figure 4 (a), (b), (c) and (d) depict the four possible growth of the modified Cascade Correlation network for each creation of hidden units. The thick arrow in figure 4 (c) and (d) indicates that there is no connection between the hidden units.

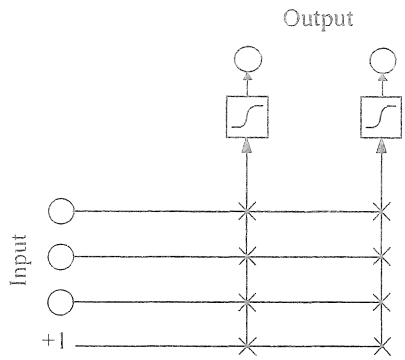


Fig. 2: Initialization of the modified Cascade Correlation network

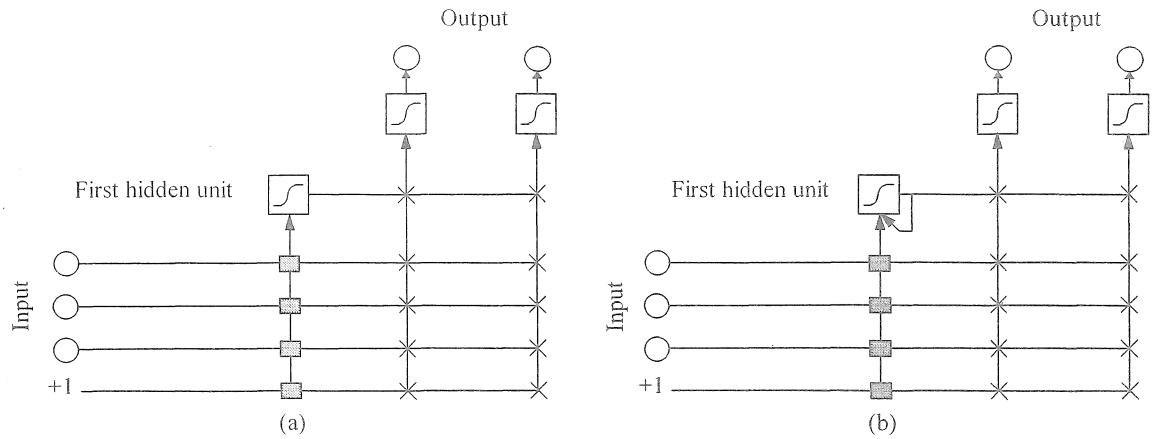


Fig. 3: First generation of the modified Cascade Correlation network:
(a): with a feedforward unit and (b): with a recurrent unit.

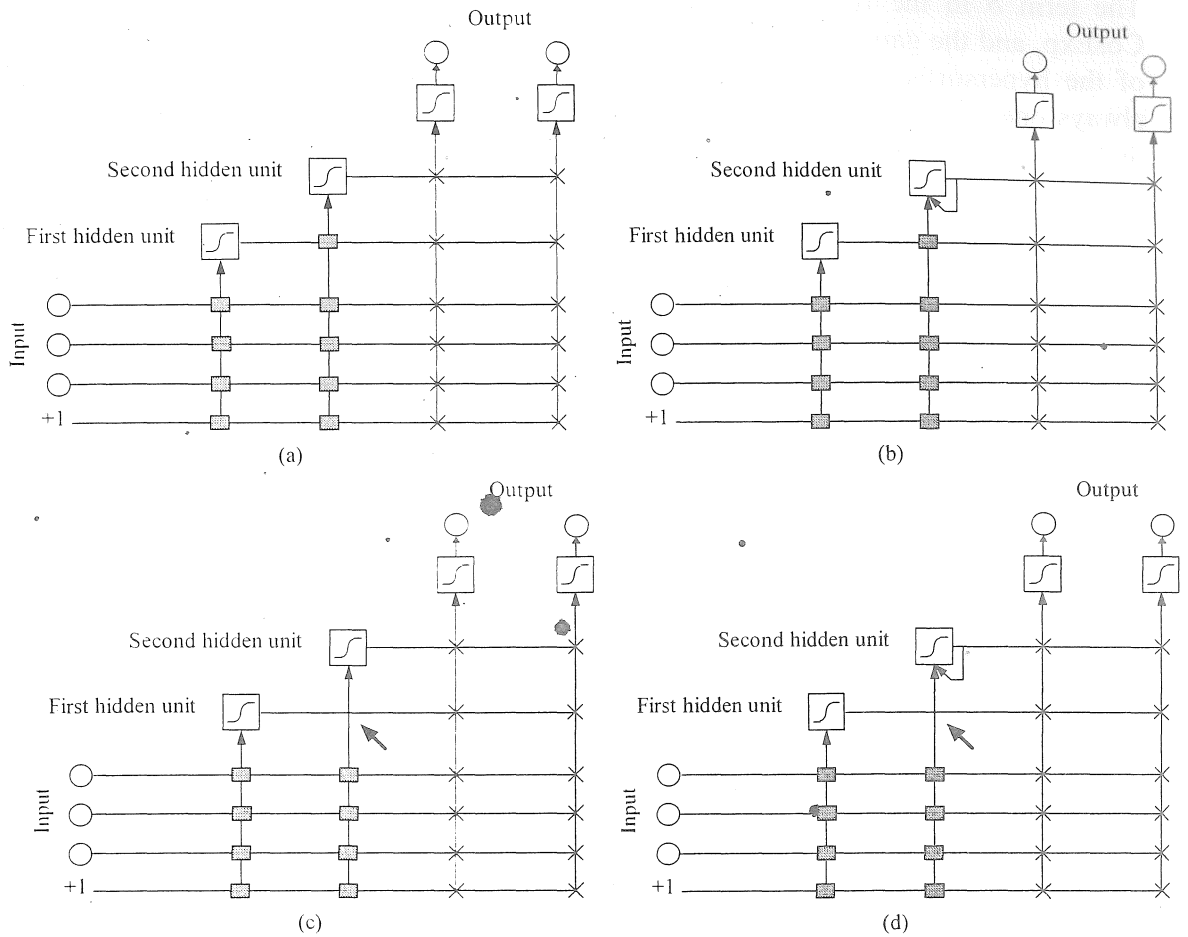


Fig. 4: Second generation of the modified Cascade Correlation network:
(a): vertical growth, feedforward, (b): vertical growth, recurrent,
(c): horizontal growth, feedforward, (d): horizontal growth, recurrent.

2 CosExp- and CosGauss activation functions

The CosExp function [6] is defined shown below:

- $f_{act}(x) = e^{-b|x-d|}\cos(c|x-d|)$. Its derivative is

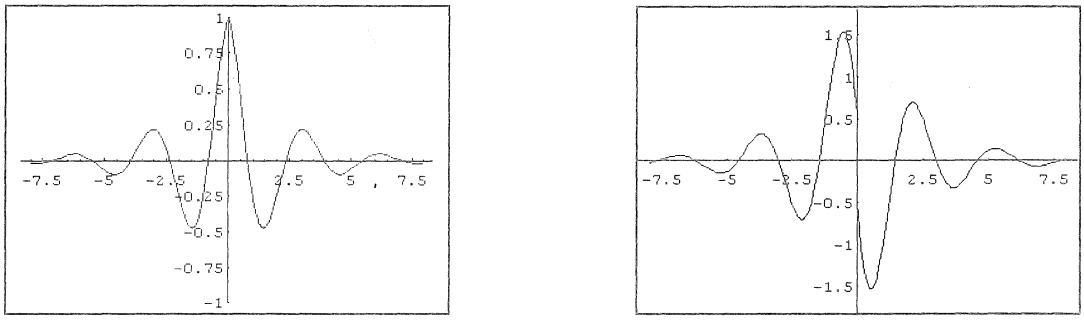
- $f'_{act}(x) = \dfrac{d-x}{|d-x|}e^{-b|x-d|}\left(b\cos(c|x-d|)+c\sin(c|x-d|)\right) \qquad (d-x \neq 0).$

The CosGauss activation function [7] and its derivative are defined as follows:

- $f_{act}(x) = e^{-b(x-d)^2}\cos(c(x-d))$.

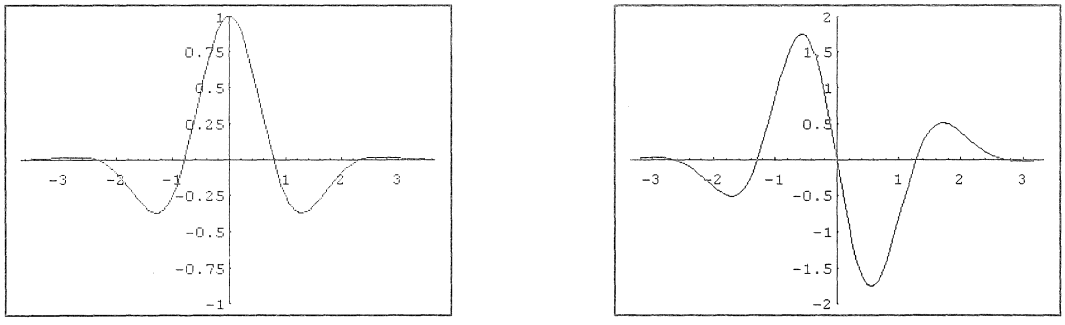
- $f'_{act}(x) = -2b(x-d)e^{-b(x-d)^2}\cos(c(x-d)) - ce^{-b(x-d)^2}\sin(c(x-d))$.

The term b in the function scales the gradient of the exponential envelope function by CosExp, and the gaussian envelope function by CosGauss, whereas d determines the position of the hypersurface. The maximum value of the CosExp- and the CosGauss functions is always one. c controls the length of the elementary periods. Observing a definite interval on the x-axis the number of the ridges depends on the value of the parameters c and b. To explain the relationship between the parameters b and c, we consider as an example, the one-dimensional case of the function. Figure 5 (a) depicts a CosExp function with parameters b=0.5, c=2 and d=0, and figure 5 (b) shows its derivative function. Figure 6 (a) depicts a CosGauss function with parameters b=0.5, c=2 and d=0, and figure 6 (b) shows its derivative function.



(a) (b)

Figure 5: (a): CosExp function with parameter b=0.5, c=2 and d=0, and (b): its derivative.



(a) (b)

Figure 6: (a): A CosGauss function with parameters b=0.5, c=2 and d=0, and (b): its derivative.

3 Experiments

The modified Cascade Correlation network using the new activation functions was tested with the well documented balance scale benchmark problem [4][5][8][9][10][11]. This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tiping to the right, tiping to the left, or being balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find out the class is to compare (left-distance * left-weight) with (right-distance * right-weight). If they are equal, it is balanced.

- Number of instances: 625 (49 balanced, 288 left, 288 right)
- Number of attributes: 4 (numeric) + class name = 5
- Attribute information:
  1. Class name: 3 (L, B, R)
  2. Left-Weight: 5 (1, 2, 3, 4, 5)
  3. Left-Distance: 5 (1, 2, 3, 4, 5)
  4. Right-Weight: 5 (1, 2, 3, 4, 5)
  5. Right-Distance: 5 (1, 2, 3, 4, 5)
- Class distribution:
  1. 46.08 percent of L
  2. 07.84 percent of B
  3. 46.08 percent of R

In the experiments the Cascade Correlation(CC), the recurrent Cascade Correlation(RCC) and the modified Cascade Correlation(M-CC) architectures are used with the following parameters:

- *candChangeThreshold* is a measure of how much correlation value of the best candidate unit must change from its previous best before this change is considered significant.
- *candDecay* is the amount that the slope of each weight coming into a candidate unit is decreased in each epoch.
- *candEpochs* is the maximum number of epochs to train the candidate units before selecting the best unit and adding it to the network.
- *candEpsilon* is the epsilon value used to train the candidate units.
- *candMu* is the maximum growth factor discussed in Fahlman's paper [1].
- *candPatience* is the number of epochs required to continue training without noticeable improvement before training is declared stagnant and stopped.
- *errorIndexThreshold* is the error index to beat when the scoring method is an index (used when continuous outputs are present). Training is stopped and victory is declared whenever error index drops below errorIndexThreshold.
- *Ncands* is the number of candidate units to place in a training pool. The best of these units will be selected to be added to the network.
- *Npools* is the number of training pools.
- *outDecay* is the amount that the slope of each weight coming into an output unit is decreased in each epoch.
- *outEpochs* is the maximum number of epochs to train the output units before training a new set of candidate units. Usually, the network will stagnate long before this occurs.
- *outEpsilon* is the epsilon value used to train the output units.
- *outErrorThreshold* is a measure of how much the error from the outputs must change from their previous best before this change is considered significant. This is used for stagnation calculation.
- *outMu* is the maximum growth factor discussed in Fahlman's paper [1].
- *outPatience* is the number of epochs to continue training without noticeable improvement before training is declared stagnant and stopped.
- *scoreThreshold* is used to designate how close a binary output has to be with respect to the correct value, before it is considered correct. The smaller this value is, the closer the network has to be to the value specified.
- *weightRange* is used to initialize the random starting weights. These values are between +/- weightRange.
- *Ntrials* is the number of networks to train on this data set.

The used parameter values are listed in table 1.

Used parameter values to test of the balance scale problem				
candChangeThreshold	candDecay	candEpochs	candEta	candMu
0.04	0.0001	30	0.0001	2.0
errorIndexThreshold	candPatience	Ncands	Npools	outDecay
0.05	25	16	1	0.0001
outPatience	outMu	outEta	outEpochs	Ntrials
50	2.0	0.0007	30	1
outErrorThreshold	scoreThreshold	weightRange		
0.01	0.5	1.0		

Table 1: Used parameter values to test the balance scale problem.

To solve the balance scale problem, we always take the sigmoid activation function as the output units of the network. The sigmoid, the tanh, the CosExp (b = 0.01, c = 1.5 and d = 0), and the CosGauss activation functions (b = 0.01, c = 1.5 and d = 0) are taken as the candidate and the hidden units. In the following, to train the modified Cascade Correlation network, the Cascade Correlation network, and the recurrent Cascade Correlation network, four candidate units in each group of the pool were installed. Initial weights were uniform random values in the range -0.1 to +0.1.

Figure 7 shows a comparison of the learning error curves between the trained Cascade Correlation network, the trained recurrent Cascade Correlation network and the trained modified Cascade Correlation network using the sigmoid activation function as the candidate and the hidden units. Figure 8 shows the topology of the trained modified Cascade Correlation network with the sigmoid activation function. The numbers indicated by the arranged hidden units present the sequential creation of the hidden units. In the following, the activation function with "*" represents the activation function as the candidate units and the activation function with "°" denotes activation function as the output unit. Figure 9 and 10 depict the results with the tanh activation function. Figure 11 and 12 show the results with the CosExp activation function. Figure 13 and 14 show the results with the CosGauss activation function.

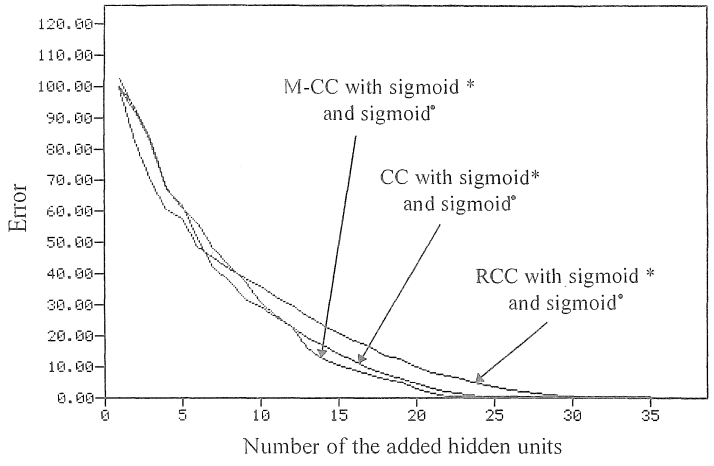


Fig 7: Comparison of the error curve between the CC-, RCC and M-CC network using the sigmoid activation function for the candidate and the hidden units.

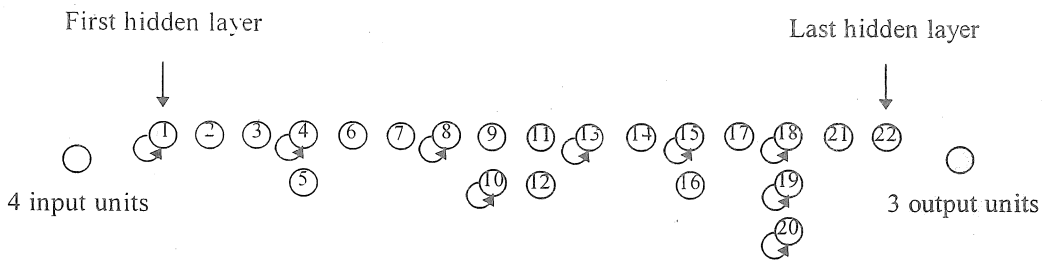


Fig. 8: Arrangement of the created hidden units on the hidden layers of the trained M-CC network using sigmoid activation function for the candidate and the hidden units.

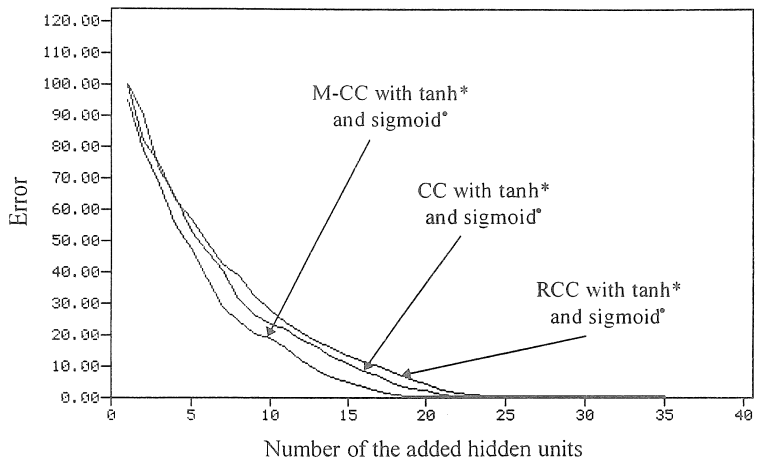


Fig 9: Comparison of the error curve between the CC-, RCC and M-CC network using the tanh activation function for the candidate and the hidden units.

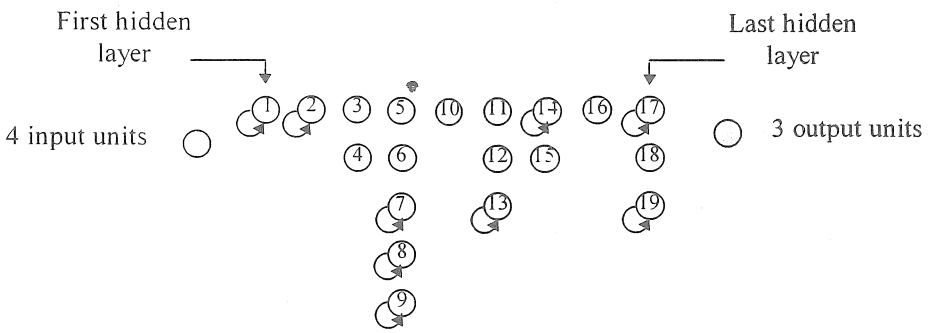


Fig. 10: Arrangement of the created hidden units on the hidden layers of the trained M-CC network using tanh activation function for the candidate and the hidden units.

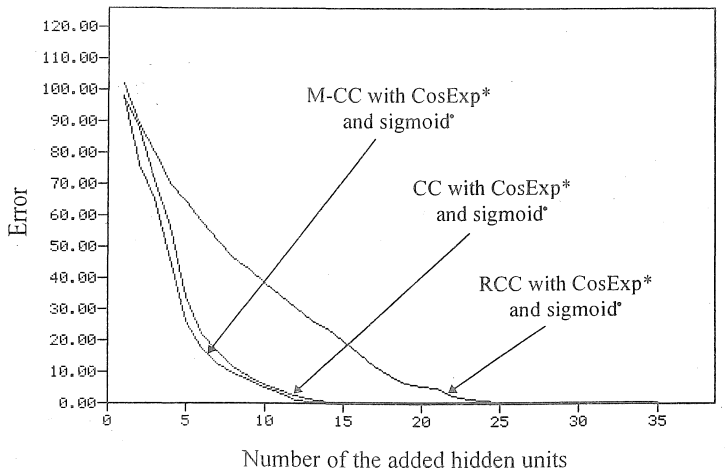


Fig 11: Comparison of the error curve between the CC-, RCC and M-CC network using the CosExp activation function for the candidate and the hidden units.

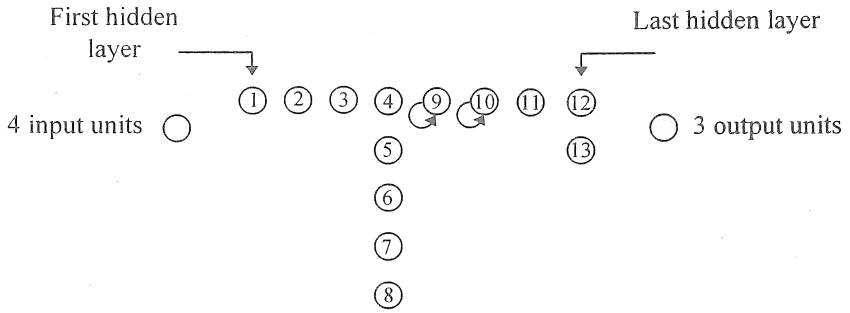


Fig. 12: Arrangement of the created hidden units on the hidden layers of the trained M-CC network using CosExp activation function for the candidate and the hidden units.

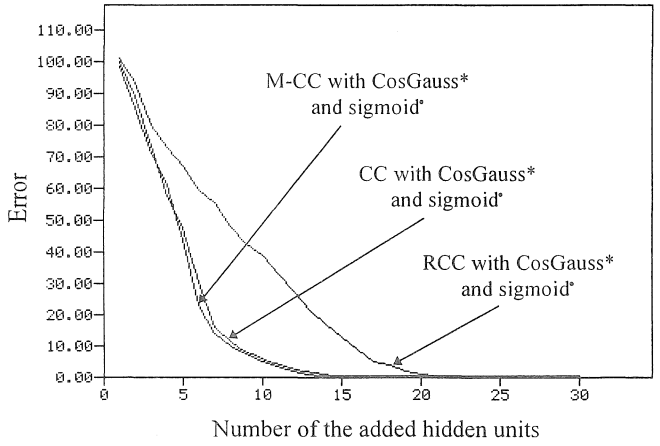


Fig 13: Comparison of the error curve between the CC-, RCC and M-CC network using the CosGauss activation function for the candidate and the hidden units.

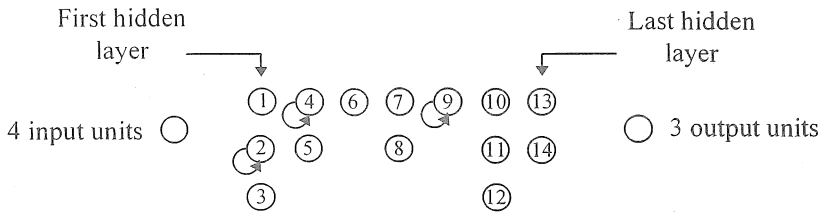


Fig. 14: Arrangement of the created hidden units on the hidden layers of the trained M-CC network using CosGauss activation function for the candidate and the hidden units.

The values of the number of created hidden layers and produced hidden units are listed in Table 2.

Network	Activation function	Number of the hidden layers	Number of the hidden units
CC	Sigmoid* and sigmoid'	24	24
RCC	Sigmoid* and sigmoid'	29	29
M-CC	Sigmoid* and sigmoid'	16	22
CC	tanh* and sigmoid'	22	22
RCC	tanh* and sigmoid'	24	24
M-CC	tanh* and sigmoid'	9	19
CC	CosExp* sigmoid'	14	14
RCC	CosExp* sigmoid'	24	24
M-CC	CosExp* sigmoid'	8	13
CC	CosGauss* sigmoid'	14	14
RCC	CosGauss* sigmoid'	21	21
M-CC	CosGauss* sigmoid'	7	14

Table 2: The number of the created hidden units and the hidden layers of the trained CC-, RCC- and M-CC network to solve of the balance scale problem.

As a test of generalization, 438 samples (70%) for training and 187 samples (30%) for testing of the 625 instances are ramdomly chosen. We ran five trials of the Cascade Correlation network, the recurrent Cascade Correlation network and the modified Cascade Correlation network on the train and the test sets. The results are as follows:

Network	Activation function	Test set	Accuracy on testing samples	Classification
CC	sigmoid * and sigmoid'	187	172.0	92.0%
RCC	sigmoid * and sigmoid'	187	168.3	90.0%
M-CC	sigmoid * and sigmoid'	187	172.0	92.0%
CC	tanh* and sigmoid'	187	173.3	92.7%
RCC	tanh* and sigmoid'	187	166.3	88.9%
M-CC	tanh* and sigmoid'	187	166.3	88.9%
CC	CosExp* and sigmoid'	187	168.0	89.8%
RCC	CosExp* and sigmoid'	187	155.0	82.9%
M-CC	CosExp* sigmoid'	187	173.6	92.8%
CC	CosGauss* sigmoid'	187	170.0	90.7%
RCC	CosGauss* sigmoid'	187	156.5	83.7%
M-CC	CosGauss* sigmoid'	187	170.7	91.3%

Table 3: Listing of the score on the test set by the balance scale problem.

3 Conclusion

In this paper, we show that the results using the modified Cascade Correlation algorithm with the CosExp- and the CosGauss activation functions as candidate units is slightly improved with respect to Cascade Correlation (with the same activation functions) and the nets are much smaller than the original Cascade Correlation architecture. The modified Cascade Correlation leads to compact and elegant network. Future work will be devoted to finding optimal parameter values of these activation functions for general purpose problems by means of evolutionary algorithms.

References

[1] Fahlman, S. E.: "*Faster-learning variations on back-propagation: An empirical study*", Proceedings of the 1988 Connectionist Models Summer School. Morgan Kaufmann, 1988.

[2] Fahlman, S. E. and Lebiere, C.: "*The cascade-correlation learning architecture*", Advances in Neural Information Processing Systems 2, Morgan Kaufmann, 1990.

[3] Fahlman, S. E.: "*The Recurrent Cascade-Correlation Architecture*", Advances in Neural Information Processing Systems 3, pp. 190-198, Morgan Kaufmann Publishers, Inc., 1991.

[4] Klahr, D., & Siegler, R.S.: "*The Representation of Children's Knowledge*", In H. W. Reese & L. P. Lipsitt (Eds.), Advances in Child Development and Behavior, pp. 61-116. New York: Academic Press, 1978.

[5] Langley, P.: "*A General Theory of Discrimination Learning*", In D. Klahr, P. Langley, and R. Neches (Eds.), Production System Models of Learning and Development, pp. 99-161. Cambridge, MA: MIT Press, 1987.

[6] Lee, S. W. and Moraga, C.: "*Neural Networks Using a Cosine-Modulated Symmetric Exponential Activation Function*", International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU '96), Granada, Spain, July, 1996.

[7] Lee, S. W. and Moraga, C.: "*A Cosine-Modulated Gaussian Activation Function for Hyper-Hill Neural Networks*", Third International Conference on Signal Processing (ICSP '96), Beijing, China, October, 1996.

[8] Newell, A.: "*Unified Theories of Cognition. Cambridge*", MA: Harvard University Press, 1990.

[9] McClelland, J. L.: "*Parallel Distibuted Processing: Implications for Cognition and Development*", Technical Report AIP-47, Department of Psychology, Carnegie-Mellon University, 1988.

[10] Siegler, R. S.: "*Three Aspects of Cognitive Development*", Cognitive Psychology, 8, pp. 481-520, 1976.

[11] Shultz, T., Mareschal, D., and Schmidt, W.: "*Modeling Cognitive Development on Balance Scale Phenomena*", Machine Learning, Vol. 16, pp. 59-88, 1994.