Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Learning the Fuzzy Connectives of a Multilayer Network Using Particle Swarm Optimization

Gaurav Parekh, *Student Member, IEEE*, and James M. Keller, *Fellow, IEEE*

***Abstract*—Fuzzy connectives provide a simple and yet a very flexible way to carry out multicriteria aggregation. Often in complex problems, the aggregation needs to be carried out at different hierarchical levels. Multilayer networks provide a natural and intuitive way for modeling such hierarchical decision making systems. In this paper we propose a novel, guided heuristic for learning the parameters of a multilayer network using particle swarm optimization. We also investigate the possibility of having multiple ways of aggregating the same information based on training data. Experiments are run by selecting several different topologies for the multilayer network. Also, a comparison is made between our method and another approach that uses back propagation for training.**

Keywords-Fuzzy connectives, multicriteria aggregation, particle swarm optimization, decision making.

I. INTRODUCTION

HUMANS are good at making decisions because we inherently learn how to associate a degree of importance with each criterion and combine them in non linear ways. But to get a computer to learn this task is quite a complex problem. Aggregating multiple criteria for the purpose of decision making has been a topic of great interest for several researchers, since it has applications in various fields including engineering, finance, medicine, etc. Several fuzzy connectives have been suggested for multicriteria aggregation [1]. The choice of the connective used is often application dependent. Generally, the connectives available are the union, intersection and mean operators. Besides these, other operators have also been suggested in literature[2][3]. These include the multiplicative hybrid operator (γ model) by Zimmermann and Zysno [4] and the OWA operator by Yager [5]. The Zimmermann's operators have been further studied in [6].

For simple decision making problems, information can be fused using one level fuzzy connectives. However for more complex problems it becomes necessary to carry out this information fusion at different hierarchical levels. We can find examples of several such problems in literature [7][8]. Multilayer networks provide a natural and intuitive way for representing such hierarchical levels.

G. Parekh is with the Electrical and Computer Engineering Department, University of Missouri, Columbia, MO 65211. (e-mail: gmppx5@mizzou.edu; phone: 480-627-9802)

J. Keller is with the Electrical and Computer Engineering Department, University of Missouri, Columbia, MO 65211. (e-mail: kellerj@missouri.edu)

There are several methods in literature for representing and aggregating information through such networks. These methods can be based on probability theory [9], Dempster-Shafer belief theory [7][10] or on fuzzy set theory [11]. In [11], Krishnapuram and Lee proposed a training algorithm to train their 'hierarchical aggregation network'. This algorithm was a variation of the back propagation algorithm and was used to arrive at the aggregation functions.

In this paper we propose a swarm intelligence approach for learning the parameters of a multilayer network containing hybrid operator activation functions. We also substantiate the possibility that alternative linguistic interpretations can result when training a network with input/output pairs only.

Sec. II gives some background information about the techniques that we have used in this paper. In Sec. III we explain our approach and the algorithm associated with it. Sec. IV presents the experimental results and a comparison with Krishnapuram and Lee's results [11]. Finally, in Sec. V, we summarize and discuss our findings and the advantages that our method has to offer.

II. BACKGROUND

A. Zimmermann's multiplicative hybrid operator

In [4], Zimmermann and Zysno proposed a hybrid operator for multicriteria aggregation that was modeled after the compensatory nature of human aggregation. This hybrid operator (γ model) can be given by the following equation:

$$Y = \left(\prod_{i=1}^{n}(a_i)^{\delta_i}\right)^{1-\gamma}\left(1-\prod_{i=1}^{n}(1-a_i)^{\delta_i}\right)^{\gamma} \qquad (2.1)$$

where, $a_i \in [0,1]$ are the criteria to be aggregated, $0 \le \gamma \le 1$ is known as the mixing coefficient and it controls the degree of compensation between the union and intersection parts of the operator, and $\sum_{i=1}^{n}\delta_i = n$ where δ_i is the weight associated with each criteria a_i and n is the number of criteria being aggregated

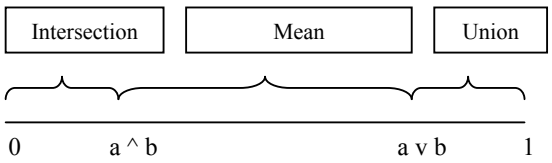

Fig. 1. Different values of γ dictate the operator being selected. The line represents values of γ going from 0 to 1.

The advantage of using this operator is that it can be operated over the entire range [0, 1]. The relation between γ and the working of the operator can be explained using Figure 1. By picking suitable values for γ we can make the Zimmerman's operator behave as an Intersection operator (γ close to 0), a Union operator (γ close to 1) or a mean operator ($0 < \gamma < 1$). In addition, it allows us to associate a weight with each criteria which is often a useful feature.

B. Particle Swarm Optimization (PSO)

The particle swarm optimization algorithm was proposed by Kennedy and Eberhart in [12]. It is a population based search algorithm that is modeled on the social behavior of birds within a flock. A swarm consists of a set of individuals (referred to as particles) that are 'flown' through high dimensional search space. Each particle represents a possible solution to the optimization problem. The position of each particle is influenced by its own experience and the experience or knowledge, of its neighbors. Each particle tries to emulate the success of other particles.

The PSO is driven by the social interaction between the particles. Individuals (particles) within the swarm learn from each other, and based on the knowledge obtained, move towards more 'promising' regions in the search space. The manner in which the particles interact is determined by the formation of neighborhoods. Different kinds of neighborhoods have been defined and studied [13][14]. In this paper, we use a star neighborhood. In the star-type neighborhood, each particle can communicate with every other particle. Hence, the best particle influences the movement of the entire swarm. Each particle tries to imitate the overall best particle and improves its own performance in the process.

The exchange of social information between particles is modeled with a velocity equation. In our case, we use the global best (*gbest*) algorithm and the velocity equation can be given as follows:

$$\vec{v}_k(t) = \vec{v}_k(t-1) + \rho_1(\vec{x}_{pbest} - \vec{x}_k(t)) + \rho_2(\vec{x}_{gbest} - \vec{x}_k(t)) \tag{2.2}$$

where, $\rho_1 = r_1 c_1$; $\rho_2 = r_2 c_2$;

$r_1, r_2 \sim U(0,1)$; $c_1 + c_2 \leq 4$

t is the iteration number and k is the particle index.

Here, ρ_1 can be thought of as the weight a particle associates with its previous experience and the knowledge it has acquired thus far and ρ_2 can be viewed as the weight a particle associates with the knowledge the entire swarm has acquired thus far.

The position of a particle in the swarm is modified by adding velocity to its current position. The equation is given by:

$$\vec{x}_k(t) = \vec{x}_k(t-1) + \vec{v}_k(t) \tag{2.3}$$

III. METHODOLOGY

In this paper, we use Zimmermann's multiplicative hybrid operator for aggregating information at every node in a decision making network. Given a set of training data (here, input criteria satisfaction and ultimate desired output), our goal is to use PSO to learn the weights and the mixing coefficients for each of the hybrid operators in the network. Thus, every particle contains all of the weights and mixing coefficients for all nodes. The sum of squared errors (SSE) is used as the fitness function. For a fixed network architecture and a training set $T = \{(\vec{a}_1, Y_1) \quad \cdots \quad (\vec{a}_N, Y_N)\}$, the algorithm that we used to train the multilayer network can be given as follows:

1. Initialize a swarm of 50 particles (number of particles was selected arbitrarily). The number of parameters in each particle, i.e., the complexity of the search space, depends on the structure of the multilayer network.
2. During initialization make sure that the restrictions $0 \leq \gamma \leq 1$ and $\sum_{i=1}^{n} \delta_i = n$ are met.
3. Set iteration count $t = 0$.

While (No convergence)

4. Evaluate the fitness of each particle.
   - 4.1 Compute the outputs for the nodes in the middle layer using equation (2.1). The values of the leaf nodes act as inputs.
   - 4.2 Once we have the values for nodes in the middle layer we can view the final node as a hypothesis and the nodes in the middle layer as sub-hypothesis that need to be aggregated to get the support for the hypothesis. Again, we use equation (2.1) to fuse the various sources of information.
   - 4.3 On obtaining the final output Y' we compute the Sum of Squared Errors (SSE) for each particle. SSE is computed over the entire data set and for any particle can be given as:

$$SSE = \sum_{k=1}^{N} (Y_k - Y_k')^2$$

   N is the number of data points.

   Our goal is to minimize the SSE.
5. Compare the performance of each particle to its personal best (*pbest*) performance so far:
   - 5.1 If fitness(current position) < fitness(*pbest*) then
     - 5.1.1 fitness (*pbest*) = fitness (current position)
     - 5.1.2 position (*pbest*) = current position
6. Compare the performance of each particle to global best particle (*gbest*) so far:
   - 6.1 If fitness(current position) < fitness(*gbest*) then
     - 6.1.1 fitness (*gbest*) = fitness (current position)
     - 6.1.2 position (*gbest*) = current position
7. Compute the new velocity for each particle in every dimension using equation (2.2).

8. Control the maximum speed with which a particle can move in any direction by setting a V_{max}. This is done to ensure that the particles don't simply fly over the 'promising' regions of the search space. (We set $V_{max} = 0.2$).
9. Update the position of each particle using equation (2.3).
10. Make sure that the restrictions $0 \le \gamma \le 1$ and $\sum_{i=1}^{n} \delta_i = n$ are met.
11. Increment the iteration count: $t = t + 1$.

End While

Convergence: We conclude that the algorithm has converged when any one of the following criteria are satisfied:

(1.) The *gbest* solution does not change for a large number of iterations.
(2.) The maximum iteration count t_{max} is reached.

We use the following scheme to ensure that the restriction $\sum_{i=1}^{n} \delta_i = n$ is met:

Let there be n criteria at a certain node that need to be aggregated. Then it is required that

$$\delta_1 + \delta_2 + ... + \delta_n = n$$

However, after updating the positions of the particles it is possible that we have:

$$\delta_1 + \delta_2 + ... + \delta_n = \Delta$$

We take care of the restriction by multiplying both sides by n/Δ:

$$\frac{n}{\Delta} \times \delta_1 + \frac{n}{\Delta} \times \delta_2 + ... + \frac{n}{\Delta} \times \delta_n = \frac{n}{\Delta} \times \Delta$$

$$\therefore \sum_{i=1}^{n} \delta_i' = n$$

where $\delta_i' = \frac{n}{\Delta} \times \delta_i$

IV. Experiments and Results

To test our method and evaluate its performance we ran several experiments using synthetically generated training and test data. We used several different topologies for the multilayer network and the results obtained from these experiments are given in Tables 1-8 under experiment 1.

For experiments 2 and 3 we used the original data of Zimmermann *et al.* in [15] and the stool/vision data from [11]. Krishnapuram and Lee used these data sets as test cases for their hierarchical aggregation network [11]. In that approach, a variation of the back propagation algorithm was used to train the network. The results obtained from their method as well as ours are shown in Tables 9-12 for comparison purposes.

Experiment 1. The general structure of the multilayer network employed for this experiment is shown in Figure 2. The structure was set by providing the number of nodes in the middle layer and the number of leaf nodes. No restriction was placed on these numbers except that they had to be greater than 0. Parameters for the multiplicative hybrid operators were randomly generated and assigned to each node. Then, a table of 1000 input values was randomly generated and corresponding outputs were calculated from successive applications of equation (2.1). The training data consisted of 800 data points and the test data had 200 data points. The results obtained are shown in Tables 1-8.

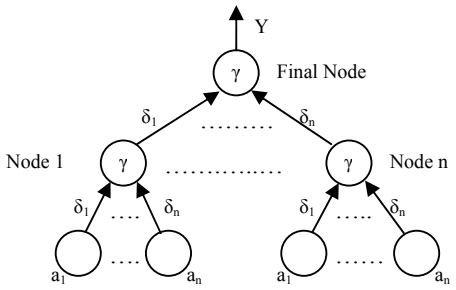

Fig. 2. General structure of the Multilayer Network used for experiment 1.

In Tables (1, 3, 5, 7) the values shown are representative samples of the training and test data sets. But the sum of squared errors (SSE) shown was that obtained over the entire training data set. From the tables it is clear that the outputs obtained after training (Y') are quite close to the target output (Y). The low SSE in training and test cases further establishes that this is true for the entire data.

In Tables (2, 4, 6, 8) we display the randomly generated parameters that we installed in each node for this experiment along with the recovered values of these parameters after training by PSO. As is evident from the results, the learned parameters are very close to the actual parameters. Table 8 provides an interesting inconsistency in the results. For node 1, the values that the PSO learned for δ_3, and δ_4 correspond to the values for δ_4, and δ_3 in the actual data. Also, interestingly the learned γ for that node dictates that the node act as a union-like operator whereas while generating the data the node acted as intersection-like operator, and yet the SSE on both training and test data is very low.

Experiment 2. For this experiment we use the original data that was used by Zimmermann and Zysno in [15]. In this data the aggregation of two criteria is carried out using a single node. Hence there are only three parameters to be learned namely (δ_1, δ_2 and γ) and there are 24 data points available. However the nature of the data is such that it is not possible to obtain a perfect match between the actual and target outputs. Thus the question is how close can we get to the target outputs. The results obtained using our method are placed side by side with the results obtained by Krishnapuram and Lee [11] (denoted as K-L method) in Tables 11 and 12. As we can see the results are almost identical.

Experiment 3. For this experiment we used the stool/vision data that Krishnapuram and Lee had used as a test case for their hierarchical aggregation network. The structure of the network is shown in Figure 3.

The four nodes going into the 'legs' criteria represent the four legs of a stool. The top of a stool can be either circular or square. Hence the two nodes going into the 'top' aggregation are actually two hypotheses, one being that the top of the stool is a parallelogram and the other being the top is an ellipse. The evidence for the stool is obtained by aggregating the 'legs' and 'top' criteria. Similarly the evidence for the 'legs' criterion is obtained by aggregating the evidence for the four separate legs and the evidence for the top is obtained by the presence of either a parallelogram or an ellipse.

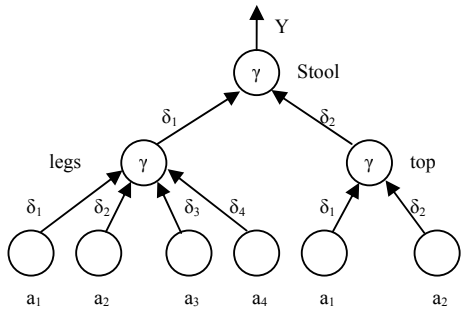

Fig. 3. Structure of the Multilayer Network for stool/vision data used for experiment 3.

Now based on intuition we can say, probably the best way to aggregate this data would be to model the 'legs' node as a mean operator, the 'top' node as a union operator, and the stool node as mean operator. This is because, based on the viewing perspective all the four legs might not be visible to the camera. Hence, we wish to model 'at least a few are visible' for the 'legs' node which can be effectively done by using a mean operator. The 'top' node can either be a parallelogram or an ellipse, therefore we need an 'OR' kind of operator i.e. a union operator. Now, if we choose the stool node to operate as an intersection operator then the support that we would get for a stool would be low if the legs were not fully seen in a view. Hence a mean connective makes a more sensible choice. These were the assumptions in [11]. However, the small amount of simulated data in Table 9 may or may not support those hypotheses.

We ran this experiment and made a few interesting observations.

(1.) If we look at the parameters that were learned by the PSO in Table 10 we find that for the 'top' node the value for δ_1 is 0 whereas the value for δ_2 is almost 2. Since the weight associated with it is 0, we can conclude that the criteria a_1 plays no role in the decision making process and can be totally neglected for that node. If we look at the data itself in Table 9, we find that the value for this criterion (a_1) always remains constant at 0.1. Hence the output for the 'top' node is solely dictated by whether a_2 is high or low. Also, since the criterion a_1 corresponds to a parallelogram we can also conclude that none of the stools that were seen had a square or a rectangular top.

(2.) Also from Table 10, we can see that contrary to our intuition the 'legs' node acts as a union operator instead of a mean operator. Also instead of associating equal weights with all the four legs the parameters that the PSO learned associate more weight with leg 1.

(3.) Also the γ that was found for the stool node using Krishnapuram and Lee's method was 0.728, which suggests a union-mean like operator. Whereas, the γ found by our method for that node is 0.365 which suggests an intersection-mean like operator.

From the above observations as well as the results in Tables 7 and 8 we get the impression that may be the parameters that can be recovered for a multilayer network are not unique and if we let the data drive the optimization process then the results obtained by the learning process may be very different than what we might expect based on our intuition. In other words, it leads us into believing that there may be multiple ways for arriving at the same conclusion.

TABLE 1. SAMPLE OF INPUT/OUTPUT DATA FOR EXPERIMENT 1 WHERE EACH NODE HAS 2 INPUTS

Sample of Training Data						
Node 1		**Node 2**		**Y**	**Y'**	**SSE**
a1	**a2**	**a1**	**a2**			
0.425	0.590	0.655	0.861	0.364	0.363	
0.768	0.452	0.629	0.668	0.209	0.211	
0.532	0.053	0.521	0.548	0.018	0.018	0.00175
0.235	0.868	0.722	0.892	0.490	0.492	
0.673	0.925	0.428	0.829	0.542	0.541	
Sample of Test Data						
0.467	0.538	0.518	0.990	0.423	0.420	
0.771	0.678	0.617	0.999	0.621	0.622	
0.810	0.344	0.392	0.109	0.005	0.005	0.00052
0.997	0.644	0.235	0.630	0.242	0.244	
0.272	0.032	0.821	0.528	0.009	0.008	

TABLE 2. THE ACTUAL AND RECOVERED PARAMETERS CORRESPONDING TO TABLE 1.

	Parameter	**Actual**	**Recovered**
Node 1	**δ1**	0.440	0.446
	δ2	1.559	1.553
	γ	0.255	0.341
Node 2	**δ1**	0.161	0.163
	δ2	1.838	1.836
	γ	0.180	0.198
Final Node	**δ1**	0.786	0.816
	δ2	1.213	1.183
	γ	0.0846	0.028

TABLE 3. SAMPLE OF INPUT/OUTPUT DATA FOR EXPERIMENT 1 WHERE THE MIDDLE LAYER HAS 3 NODES, EACH WITH 2 LEAF NODES

Sample of Training Data								
Node 1		Node 2		Node 3		Y	Y'	SSE
a1	a2	a1	a2	a1	a2			
0.639	0.497	0.772	0.864	0.074	0.819	0.812	0.810	
0.069	0.985	0.341	0.527	0.465	0.980	0.708	0.706	
0.618	0.240	0.999	0.344	0.274	0.785	0.531	0.528	0.007383
0.325	0.189	0.962	0.829	0.842	0.094	0.488	0.489	
0.820	0.004	0.537	0.581	0.007	0.940	0.463	0.459	
Sample of Test Data								
0.428	0.463	0.616	0.456	0.902	0.791	0.621	0.621	
0.280	0.669	0.344	0.738	0.230	0.476	0.647	0.642	
0.667	0.625	0.045	0.163	0.905	0.061	0.281	0.284	0.002147
0.057	0.847	0.657	0.218	0.941	0.720	0.459	0.458	
0.710	0.457	0.158	0.717	0.221	0.424	0.625	0.624	

TABLE 4. THE ACTUAL AND RECOVERED PARAMETERS CORRESPONDING TO TABLE 3.

	Parameter	Actual	Recovered
Node 1	δ1	0.863	0.887
	δ2	1.136	1.112
	γ	0.789	0.791
Node 2	δ1	0.219	0.213
	δ2	1.780	1.786
	γ	0.263	0.208
Node 3	δ1	0.189	0.202
	δ2	1.810	1.797
	γ	0.067	0.082
Final Node	δ1	1.096	1.094
	δ2	1.255	1.235
	δ3	0.648	0.669
	γ	0.856	0.861

TABLE 5. SAMPLE OF INPUT/OUTPUT DATA FOR EXPERIMENT 1 WHERE THE MIDDLE LAYER HAS 2 NODES, EACH WITH 3 LEAF NODES

Sample of Training Data								
Node 1			Node 2			Y	Y'	SSE
a1	a2	a3	a1	a2	a3			
0.709	0.897	0.832	0.223	0.990	0.228	0.637	0.638	
0.901	0.467	0.771	0.492	0.470	0.599	0.668	0.668	
0.392	0.794	0.713	0.402	0.791	0.618	0.374	0.374	0.000693
0.325	0.643	0.955	0.497	0.272	0.729	0.329	0.328	
0.538	0.787	0.884	0.453	0.169	0.339	0.400	0.398	
Sample of Test Data								
0.586	0.992	0.687	0.945	0.704	0.361	0.568	0.567	
0.963	0.245	0.999	0.487	0.636	0.652	0.685	0.685	
0.653	0.290	0.051	0.576	0.311	0.831	0.263	0.263	0.000195
0.859	0.340	0.137	0.849	0.292	0.290	0.401	0.401	
0.638	0.121	0.231	0.363	0.280	0.593	0.266	0.266	

TABLE 6. THE ACTUAL AND RECOVERED PARAMETERS CORRESPONDING TO TABLE 5.

	Parameter	Actual	Recovered
Node 1	δ1	1.971	1.974
	δ2	0.462	0.461
	δ3	0.565	0.563
	γ	0.325	0.270
Node 2	δ1	1.028	1.019
	δ2	0.613	0.618
	δ3	1.357	1.362
	γ	0.966	0.966
Final Node	δ1	0.994	0.974
	δ2	1.005	1.025
	γ	0.403	0.433

TABLE 7 SAMPLE OF INPUT/OUTPUT DATA FOR EXPERIMENT 1 WHERE THE MIDDLE LAYER HAS 2 NODES, ONE OF THEM HAS 4 LEAF NODES AND THE OTHER HAS 3 LEAF NODES.

Sample of Training Data									
Node 1				Node 2			Y	Y'	SSE
a1	a2	a3	a4	a1	a2	a3			
0.981	0.375	0.713	0.239	0.496	0.742	0.375	0.514	0.521	
0.237	0.420	0.114	0.494	0.462	0.430	0.784	0.282	0.280	
0.906	0.702	0.665	0.863	0.612	0.961	0.408	0.745	0.743	0.0311
0.066	0.402	0.070	0.021	0.119	0.731	0.437	0.226	0.224	
0.847	0.182	0.674	0.408	0.865	0.938	0.177	0.801	0.808	
Sample of Test Data									
0.942	0.523	0.371	0.356	0.326	0.862	0.347	0.488	0.488	
0.402	0.003	0.273	0.880	0.074	0.469	0.696	0.104	0.103	
0.391	0.628	0.822	0.374	0.846	0.201	0.331	0.247	0.244	0.0095
0.265	0.609	0.044	0.276	0.019	0.221	0.820	0.017	0.017	
0.657	0.712	0.154	0.906	0.892	0.139	0.308	0.200	0.190	

TABLE 8. THE ACTUAL AND RECOVERED PARAMETERS CORRESPONDING TO TABLE 7.

	Parameter	Actual	Recovered
Node 1	δ1	0.544	0.549
	δ2	0.759	0.877
	δ3	0.998	1.694
	δ4	1.697	0.879
	γ	0.343	0.801
Node 2	δ1	1.132	1.127
	δ2	1.769	1.774
	δ3	0.098	0.098
	γ	0.788	0.758
Final Node	δ1	0.022	0.067
	δ2	1.977	1.932
	γ	0.067	0.162

TABLE 9. THE OUTPUTS FOR STOOL/VISION DATA USING OUR METHOD AND KRISHNAPURAM AND LEE'S METHOD.

Node 1				Node 2		Target (Y)	Our Method (Y')	K-L Method (Y')	SSE
a1	a2	a3	a4	a1	a2				
0.1	0.1	0.1	0.1	0.1	0.1	0.01	0.081	0.058	
0.1	0.1	0.1	0.1	0.1	0.9	0.3	0.274	0.267	
0.1	0.1	0.1	0.9	0.1	0.1	0.1	0.127	0.109	
0.1	0.1	0.1	0.9	0.1	0.9	0.4	0.388	0.396	Our Method = 0.0125
0.1	0.1	0.9	0.9	0.1	0.1	0.2	0.181	0.173	
0.1	0.1	0.9	0.9	0.1	0.9	0.5	0.508	0.499	K-L Method = 0.073
0.1	0.9	0.9	0.9	0.1	0.1	0.3	0.241	0.29	
0.1	0.9	0.9	0.9	0.1	0.9	0.6	0.63	0.638	
0.9	0.9	0.9	0.9	0.1	0.1	0.4	0.423	0.516	
0.9	0.9	0.9	0.9	0.1	0.9	0.99	0.972	0.99	

TABLE 10. THE PARAMETERS RECOVERED USING OUR METHOD AND KRISHNAPURAM AND LEE'S METHOD FOR STOOL/VISION DATA.

	Parameter	Recovered Our Method	Recovered K-L Method
'legs' Node	δ1	2.999	1
	δ2	0.548	1
	δ3	0.285	1
	δ4	0.166	1
	γ	0.925	0.692
'top' Node	δ1	0.000	Not reported
	δ2	1.999	Not reported
	γ	1	0.885
Stool Node	δ1	1.210	0.998
	δ2	0.789	1.002
	γ	0.365	0.728

TABLE 11. THE OUTPUTS FOR ZIMMERMANN DATA USING OUR METHOD AND KRISHNAPURAM AND LEE'S METHOD.

Zimmermann Data				
Node 1		Target Output (Y)	Our Method (Y')	K-L Method (Y')
a1	a2			
0.426	0.241	0.215	0.268	0.269
0.352	0.662	0.427	0.494	0.494
0.109	0.352	0.221	0.170	0.171
0.63	0.052	0.212	0.162	0.163
0.484	0.496	0.486	0.466	0.466
0	0	0	0	0
0.27	0.403	0.274	0.294	0.3
0.156	0.13	0.119	0.091	0.091
0.79	0.284	0.407	0.460	0.462
0.725	0.193	0.261	0.352	0.354
1	1	1	1	1
0.33	0.912	0.632	0.621	0.621
0.949	0.02	0.247	0.156	0.157
0.202	0.826	0.5	0.474	0.474
0.744	0.551	0.555	0.632	0.633
0.572	0.691	0.585	0.635	0.635
0.041	0.975	0.355	0.305	0.304
0.534	0.873	0.661	0.724	0.724
0.674	0.587	0.57	0.621	0.622
0.44	0.45	0.418	0.413	0.433
0.909	0.75	0.789	0.833	0.834
0.856	0.091	0.303	0.283	0.285
0.974	0.164	0.515	0.423	0.425
0.073	0.788	0.324	0.310	0.309
SSE Our Method = 0.0628			**SSE K-L Method = 0.064**	

TABLE 12. THE PARAMETERS RECOVERED USING OUR METHOD AND KRISHNAPURAM AND LEE'S METHOD FOR ZIMMERMANN DATA.

Node 1	Parameter	Recovered Our Method	Recovered K-L Method
	δ1	0.886	0.889
	δ2	1.114	1.111
	γ	0.587	0.589

V. CONCLUSIONS

In this paper we proposed a novel method for learning the fuzzy connectives of a multilayer network using particle swarms. The method proposed by us provides several advantages:

1) Since the γ model is used there is no switching required between union, intersection and mean operators.
2) The particle swarms are driven by simple velocity and position equations hence there is no complex differentiation involved.
3) It provides a simpler and better guided heuristic for learning the fuzzy connectives than Krishnapuram and Lee's hierarchical aggregation network that used a variation of the back propagation algorithm by overcoming all the drawbacks, namely:
    a. The connective type could change during the iterative process (they too used the γ model to overcome this drawback).
    b. Since the method was based on back propagation algorithm with multiplicative hybrid operator activation functions, it involved the computation of complex derivatives.
    c. There was a high convergence time.
    d. The aggregation functions were highly nonlinear and sometimes involved clipping.

Also, we showed that based on the data it is possible to find multiple ways of arriving at the same output from given inputs. Hence the parameters of a multilayer network are not always unique. Sometimes the solution found by the optimization process may be totally contrary to our intuition and more often than not, such solutions provide us with a better performance than what we initially hoped.

REFERENCES

[1] D. Dubois and H. Prade, "A review of fuzzy set aggregation connectives," *Inform. Sci.*, vol. 36, pp. 85–121, 1985.

[2] B. Bouchon-Meunier, "Aggregation and Fusion of Imperfect Information," *Studies in Fuzziness and Soft Computing*, vol. 12, Physica Verlag, Springer, 1998.

[3] T. Calvo, A. Kolesarova, M. Komornikova, and R. Mesiar, "Aggregation operators: properties, classes and construction methods," *Aggregation Operators New Trends and Applications*, pp. 3--104. Physica-Verlag, Heidelberg, New York, 2002.

[4] H. J. Zimmermann and P. Zysno, "Decision and evaluations by hierarchical aggregation of information," *Fuzzy Sets and Systems*, vol. 10, pp. 243–260, 1983.

[5] R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making," *IEEE Trans. Systems Man Cybernet.*, vol. 18, pp. 183–190, 1988.

[6] A. Pradera, E. Trillas, and T. Calvo, "A general class of triangular norm-based aggregation operators: quasi-linear T-S operators," *Internat. J. Approx. Reason.*, vol. 30 (1), pp. 57--72, 2002.

[7] J. Gordan and E. J. Shortliffe, "A method for managing identical reasoning in a hierarchical hypothesis space," *Artificial Intelligence*, vol. 26, pp. 323–357, 1985.

[8] Z. Li, "Uncertainty management in a pyramid vision system," *Internat. J. Approx. Reason.*, vol. 3, pp. 59–85, 1989.

[9] J. Pearl, "Fusion, propagation and structuring in belief networks" *Artificial Intelligence*, vol. 29, pp. 241–288, 1986.

[10] R. J. Krishnapuram, "A belief maintenance scheme for hierarchical knowledge-based image analysis systems," *Proc. 3rd IFSA Congress*, Seattle, pp. 333–336, 1989.

[11] R. J. Krishnapuram and J. Lee, "Fuzzy-connective based hierarchical aggregation networks for decision making," *Fuzzy Sets and Systems,* vol. 46, pp. 11–27, 1992.

[12] J. Kennedy, R. Eberhart, "Particle Swarm Optimization," *IEEE Int'l Conf. On Neural Networks*, vol. 4, pp. 1942-1948, 1995.

[13] J. Kennedy, "Small Worlds and Mega Minds: Effect of neighborhood topology on particle swarm performance," *IEEE Congress on Evolutionary Computation*, vol. 3, pp. 1931–1938, 1999.

[14] J. Kennedy, R. Eberhart, "The particle swarm: social adaptation in information-processing systems," *New Ideas in Optimization*, McGraw-Hill, pp. 379–387, 1999.

[15] H. J. Zimmermann and P. Zysno, "Latent connectives in human decision making," *Fuzzy Sets and Systems*, vol. 4, pp. 37–51, 1980.