01 Jan 2006

Density Estimation Using a Generalized Neuron

R. Kiran

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

M. Palaniswami

Density Estimation Using a Generalized Neuron

Raveesh Kiran and Ganesh K. Venayagamoorthy
Real-Time Power and Intelligent Systems Lab.
Dept. of Electrical and Computer Engineering
University of Missouri - Rolla, MO 65409, U.S.A
rkktb@umr.edu and gkumar@ieee.org

Marimuthu Palaniswami
Dept. of Electrical and Electronic Engineering
The University of Melbourne, Victoria-3010,
Australia
swami@ee.mu.oz.au

Abstract - Neural Networks have been shown to be useful tools for density estimation. However, the training of neural network structures is time consuming and requires fast processors for practical applications. A new method with a Generalized Neuron (GN) for density estimation is presented in this paper. The GN is trained with the particle swarm optimization algorithm which is known to have fast convergence than the standard backpropagation algorithm. Results are presented to show that the GN can estimate the density functions for distribution functions with different means and variances. This density estimation method can also be applied to the multi-sensor data fusion process.

Keywords: Density Estimation, generalized neuron, particle swarm optimization, probability distribution function.

1 Introduction

A frequently encountered problem in the statistical process control (SPC) is the estimation of the probability density function or simply the density function of variables. A majority of the problems encountered in the world of science and engineering deal with large amounts of data which need to be modeled in a probabilistic manner. A number of phenomena have a good amount of complexity present in them. In most of such cases a probabilistic formulation seems to be the only feasible solution from the computational view. Some of the practical applications of the density estimation includes, signal denoising and edge-preserving [1], forecasting of the power distribution [2], biomedical application like current density within a heart [3], etc. Density estimation based data fusion has applications in various tasks such as object tracking, 3D modeling, motion analysis, robot localization and in various other military and non-military applications [4], [5].

Neural networks based methods have also been proposed for estimation of the density function [6]. Some of the disadvantages of the conventional neural networks, like the one used in [6], are larger training time and more memory and hardware constraints. This paper proposes a new method for the estimation of the density function using the Generalized Neuron (GN) trained offline with the Particle Swarm Optimization (PSO). The probability distribution function is first approximated by a GN and then the density function is obtained by differentiating the GN function.

A Generalized Neuron (GN) trained with the Backpropagation (BP) learning algorithm has been shown to overcome some of the drawbacks with conventional feedforward neural networks such as the multilayer perceptrons [7]. Some of the advantages of the GN when compared to the conventional neural network are that there are lesser requirements in terms of memory and speed for hardware implementation. The training time for the weights can be reduced by reducing the number of unknown parameters (weights) to be determined.

Particle swarm optimization technique, which is based on the behavior of a flock of birds or school of fish, is a kind of evolutionary computation [8], [9]. It has been shown previously that the PSO training algorithm has better convergence in fewer computations than BP algorithm for training neural networks to achieve the same performance [10]. In this paper, the GN for the probability distribution function approximation is trained using the PSO training algorithm.

The paper is organized as follows: In section 2, the architecture of the GN considered in this paper is explained. In section 3, a brief overview of the PSO technique is given. Section 4 deals with the application of the PSO to training a GN. The concept of estimation of a distribution function and density using the GN has been discussed in section 5. Section 6 represents some simulation results first to show that a GN can be differentiated to obtain a derivative of the function approximated by a GN and then density estimation results are obtained for Gaussian distributions with different mean and variance. Section 7 discusses about application of the density estimation for multi-sensor data fusion. Finally, a brief conclusion is given in section 8.

2 Generalized Neuron

The general structure of the common neuron model is an aggregation function and a thresholding function. The general neural network model consists of three distinct layers namely the input layer, the hidden layer and the output layer. Each of these layers consists of a number of

simple neurons that are interconnected. There may be more than one hidden layer in cases involving more complex problems. Also the number of neurons in each layer depends on the type of application it is being used for. Thus it can be seen that as the complexity of the problem increases, the number of neurons and the number of weights to be found also tends to increase. Although the aggregation operators are generally crisp, they overlook the fact that most of the processing in the neural networks is done with incomplete information at hand. The GN model uses partly sum and partly product to take into account the vagueness involved, thus over coming such drawbacks.

The use of a sigmoidal thresholding function and an ordinary product or summation aggregation in the simple neuron model does not always give satisfactory results. This is because real life problems generally involve some amount of nonlinearity. Hence the GN, which has both sigmoidal and the Gaussian functions with weight sharing, can be used to overcome such problems. Due to this the GN has more flexibility and the ability to cope better with the nonlinearity involved in any application.

Unlike the common neuron model which has either $\prod$ (product) or $\sum$ (summation) aggregation function, the GN model has both $\sum$ and $\prod$ aggregation functions.

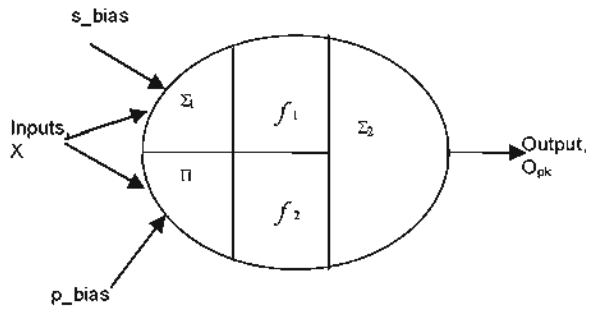

Figure 1: Generalized neuron model

The sigmoidal characteristic function (f_1) is used with the $\sum_1$ summation aggregation function while the Gaussian characteristic function (f_2) is used with the $\prod$ product aggregation function. Thus, it can be seen that flexibility at both the aggregation and the threshold level is present in the GN and so it is better equipped to cope with the nonlinearities involved in any type of application.

The output of the $\sum_1$ part with the sigmoidal activation function for f_1 of the GN is as shown below [5]:

$$O_\Sigma = f_1(s\_net) = \frac{1}{1+e^{(-\lambda_s \times s\_net)}} \qquad (1)$$

where,

$$s\_net = \sum W_{\Sigma i} X_i + X_{o\Sigma} \qquad (2)$$

and λ_s is the gain of $\sum_1$ part.

The output of the $\prod$ part with the Gaussian activation function for f_2 of the GN is as shown below:

$$O_\pi = f_2(pi\_net) = e^{(-\lambda_p \times pi\_net^2)} \qquad (3)$$

where,

$$pi\_net = \prod W_{\pi i} X_i \times X_{o\pi} \qquad (4)$$

and λp is the gain of $\prod$ part.

The final output (O_{pk}) of the neuron is a function of the two outputs O_Σ and O_π with the weights W and $(1\text{-}W)$, respectively, and can be written in the mathematical form as

$$O_{pk} = O_\pi \times (1 - W) + O_\Sigma \times W \qquad (5)$$

Thus, it can be seen from the GN model here, that there is only one output. If more than one output is required then one GN is used for each output. The number of weights in case of the GN is equal to twice the number of inputs plus one. This is very much lower when compared to the number of weights in a multilayer feedforward neural network. By reducing the number of unknown weights, the training time can be reduced.

In this model summation and product are the aggregations function used. But other fuzzy operators such as the max, min and the compensatory operators can also be used. Unlike the thresholding functions like the sigmoidal and the Gaussian functions used here, other functions like sine, cosine, hyperbolic tangent, linear functions, etc can also be used.

It can also be seen that the outputs of the sigmoidal and the Gaussian functions are summed up and hence this type of model is called as the summation type generalized neuron model. Similarly, if the product of the sigmoidal and the Gaussian functions is used, then that type of model is called as the product type generalized neuron model. It has been found that the GN model works well in most of the cases.

3 Particle Swarm Optimization

Particle Swarm Optimization is a type of evolutionary computing technique. The PSO algorithm is a population-based search algorithm, based on the simulation of the social behavior of birds within a flock. A swarm consists of a set of particles, where each particle represents a potential solution. The changes to the position of a particle and its operation in a swarm are influenced by the experience and the knowledge of its neighbors.

Initially a set of random solutions or a set of particles are considered. A random velocity is given to each particle and they are flown through the problem space. Each particle has memory which is used to keep track of the previous best position and corresponding fitness. The best value of the position of each individual is stored as 'p_{id}'. In other words, 'p_{id}' is the best position acquired by an individual particle during the course of its movement within the swarm. It has another value called the 'p_{gd}', which is the best value of all the particles 'p_{id}' in the swarm. The basic concept of the PSO technique lies in

accelerating each particle towards its 'p_{id}' and 'p_{gd}' locations at each time step. The figure below briefly illustrates the concept of PSO where

- x_{id} *(k)* is the current position.
- x_{id} *(k+1)* is the modified position.
- v_{id} *(k)* is the initial velocity
- v_{id} *(k+1)* is the modified velocity.

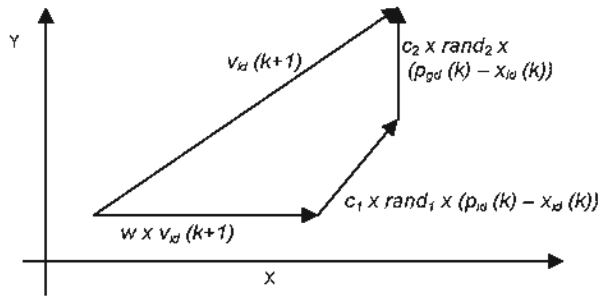

Figure 2: PSO particle update process illustrated for a two dimensional case

i). Initialize a population of particles with random positions and velocities in the problem space.
ii). For each particle, evaluate the desired optimization fitness function.
iii). Compare the particles fitness evaluation with the particles p_{id} If current value is better than the p_{id} then set p_{id} value equal to the current location.
iv). Compare the best fitness evaluation with the population's overall previous best. If the current value is better than the p_{gd}, then set p_{gd} to the particle's array and index value.
v). Update the particle's velocity and position according to the equations shown below:

The velocity of the i^{th} particle of d dimension is given by:

$$v_{id}(k+1) = w \times v_{id}(k) + c_1 \times rand_1 \times (p_{id}(k) - x_{id}(k)) + c_2 \times rand_2 \times (p_{gd}(k) - x_{id}(k)) \quad (6)$$

The position vector of the i^{th} particle of d dimension is changed as follows:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (7)$$

vi). Repeat the step ii) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations or epochs.

PSO has many parameters and these are described as follows:

- w - Inertia weight, controls the exploration and exploitation of search space, dynamically adjusts velocity
- V_{max} - Maximum velocity for the particles.
- V_{id} - Velocity of i^{th} particle with d dimensions
- X_{id} - Position of i^{th} particle with d dimensions
- c_1 - Cognition constant
- c_2 - Social constant

In case the velocity of the particle exceeds V_{max} then it is reduced to V_{max}. Thus, the resolution and fitness of search depends on the V_{max}. If V_{max} is too high, then particles will move in larger steps and so the solution reached may not be the as good as expected. If V_{max} is too low, then particles will take a long time to reach the desired solution.

4 IMPLEMENTATION OF PSO FOR GN TRAINING

Selection of the PSO parameters plays an important role in the optimization of any problem. The selection of the parameters of the GN also plays an important role is getting good results. The PSO parameters and the parameters of the GN are determined by trail and error experimentation. The value of the maximum velocity and the search space limitation depends on the type of problem it is being applied to. The number of particles was also varied. But it was observed that a very high value of the particles lead to an increase in the computational time. Hence a compromise between the computational time and the performance needs to be taken. Batch training is used in this paper.

The following set of parameters is used for the calculations in this paper.

- Maximum velocity, V_{max} 2
- Maximum search space range (-100,100)
- Inertia weight, W 0.8
- Acceleration constants, c_1 c_2 2, 2
- Size of swarm 25

The parameters of the GN are also varied over a wide range and it is found that the value of the parameters is problem dependent. So the values of the parameters are defined separately for each problem. The values of the gain scale factors (λ_s, λ_p) in the GN are determined using trail and error.

5 ESTIMATION OF DENSITY FUNCTION

Assume a sample of m observations x_1, $x_2 \ldots x_n$ whose underlying density function, f(x), is to be estimated. Let F(x) be the corresponding distribution function. Without loss of generality, the observations are assumed to be sorted in such a way that $x_1 \le x_2 \le \ldots\ldots \le x_m$. For the problem of the density estimation the basic idea is that if a continuous estimation of the distribution function $\hat{F}(x)$ is available, then the estimation of the density function could be obtained by differentiating $\hat{F}(x)$,

$$\hat{f}(x) = \hat{F}'(x). \quad (8)$$

Sigmoidal function and the Gaussian function have been used along with the two aggregation functions. If the input to the neural network is x, then the output of the neural network is

$$N(x) = O_\pi \times (1 - W) + O_\Sigma \times W \qquad (9)$$

where,

$$O_\Sigma = \frac{1}{1+e^{(-\lambda_s \times s\_net)}}$$

$$s\_net = \sum W_\Sigma X_i + X_{o\Sigma}$$

$$O_\pi = e^{-(\lambda_p \times pi\_net^2)}$$

$$pi\_net = \prod W_{\pi i} X_i \times X_{o\pi}$$

It is reasonable to take N(x) as the estimation of the distribution function, i.e.

$$\hat{F}(x) = N(x) \qquad (10)$$

Obviously, $\hat{F}(x)$ is of continuous form so that the equation (8) and equation (10) we have,

$$\hat{f}(x) = N'(x)$$

$$= dO_\pi \times (1 - W) + dO_\Sigma \times W \qquad (11)$$

where,

$$dO_\pi = - O_\pi \times \lambda_p \times 2 \times x(i) \times W_\pi^2$$

$$dO_\Sigma = O_\Sigma \times (1 - O_\Sigma) \times \lambda_s \times W_\Sigma$$

6 RESULTS

6.1 Nonlinear Static Function

In order to test the proper functioning of the GN, it is trained on the non-linear quadratic equation $y = 2x^2 + 1$, with the data points in the range of (-1, 1). The values of λ_s and λ_p are taken to be 10 and 1 respectively. The result shown in Figure 3 is obtained from the GN after it is run for 500 epochs. It has a Mean Square Error (MSE) of 4.97×10^{-5}. Next the concept of differentiation is applied to this function and the results are obtained. Figure 4 shows the differentiated output obtained for the function $y = 2x^2+1$ which is $dy/dx = 4x$ and it has a MSE of 0.0017.

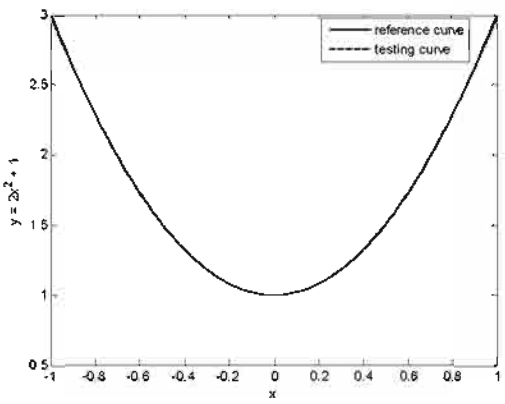

Figure 3: Output for $y = 2x^2+1$ with GN trained with PSO for 500 epochs

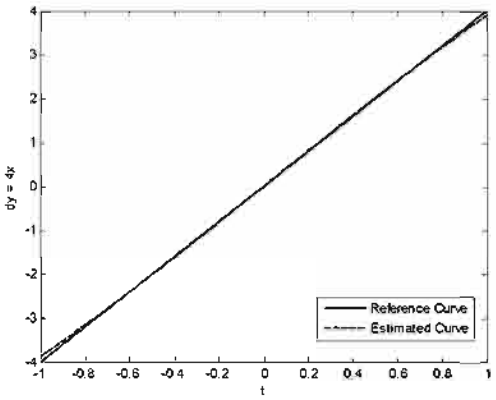

Figure 4: Differentiated output $dy/dx = 4x$

6.2 Time Varying Function

The GN is now applied to a sinusoidal function, $x = sin(2\pi ft)$, where 't' is varied in the range of (0, 1). The values of λ_s and λ_p are taken to be 0.7 and 1 respectively. Figure 5 is obtained from the GN after it is run for 500 epochs. The MSE value is found to be 5.61×10^{-7}. The result shown in Figure 6 is for the differentiated output obtained from the sinusoidal function which is $dy/dx = 2\pi f \cos(2\pi ft)$. The MSE is found to be 3.49×10^{-4}.

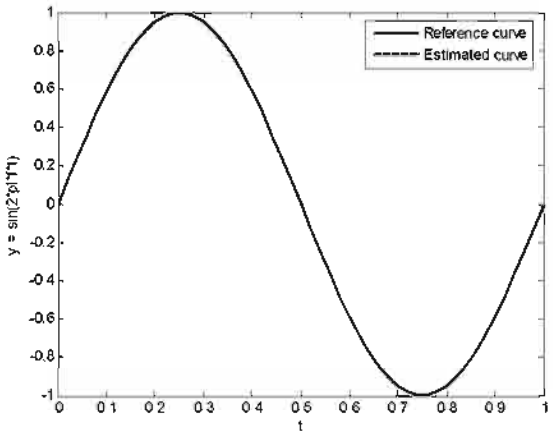

Figure 5: Output for $y = sin(2\pi ft)$ with GN trained with PSO for 500 epochs

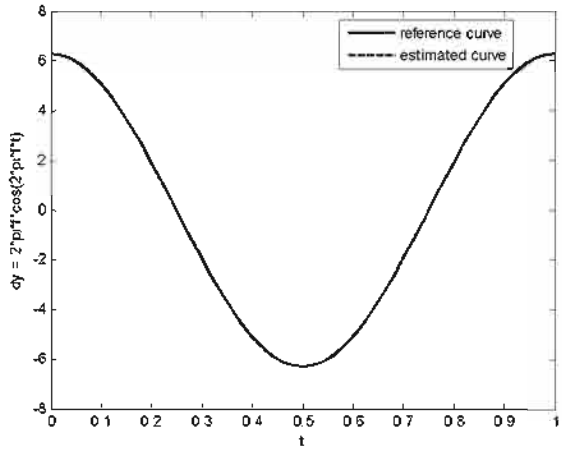

Figure 6: Differentiated output $dy/dx = 2\pi f \cos(2\pi ft)$

6.3 Gaussian Distribution

It can be shown from the following results that good approximations, for the following functions F(x) and f(x), can be obtained from N(x) and N'(x) respectively.

$$F(x) = \frac{1}{\sqrt{2\pi\sigma}} \int_{-\infty}^{x} \exp(-\frac{(t-\mu)^2}{2\sigma^2})dt \qquad (12)$$

and

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(t-\mu)^2}{2\sigma^2}) \qquad (13)$$

where,

μ = mean of the distribution

σ^2 = variance of the distribution

The real Gaussian distribution obtained after 1000 epochs is shown in Figure 7, for a $\mu = 0$ and $\sigma^2 = 1$, while the real Gaussian density function is shown in Figure 8. The former has a MSE of 1.65×10^{-5} while the latter has an MSE of 1.03×10^{-4}. The values of λ_s and λ_p are taken to be 0.7 and 1 respectively.

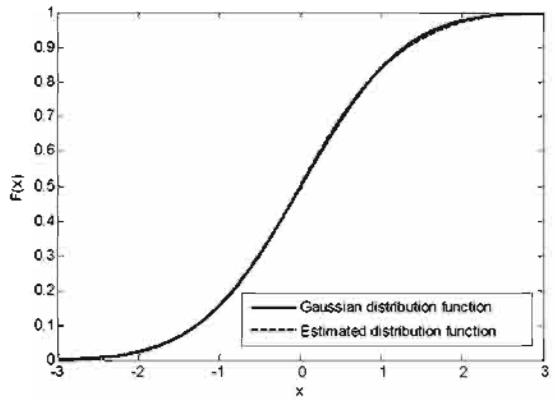

Figure 7 : Gaussian distribution function vs. estimated distribution function ($\mu = 0$ and $\sigma^2 = 1$)

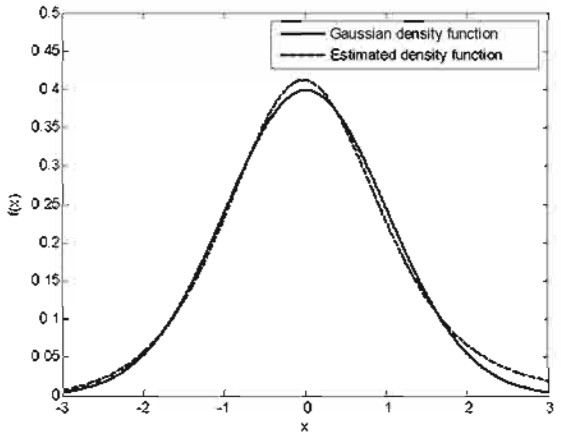

Figure 8: Gaussian density function vs. estimated density function ($\mu = 0$ and $\sigma^2 = 1$)

Figure 9 shows the results for the Gaussian distribution function with a $\mu = 0.5$ and $\sigma^2 = 1$. It has a MSE of 2.62×10^{-5} for 1000 epochs. The Gaussian density function result is shown in Figure 10. It has a MSE value of 1.05×10^{-4}. The values of λ_s and λ_p are taken to be -0.7 and 1 respectively.

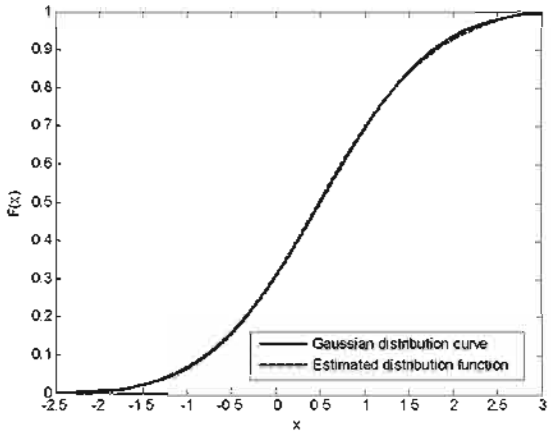

Figure 9 : Gaussian distribution function vs. estimated distribution function ($\mu = 0.5$ and $\sigma^2 = 1$)

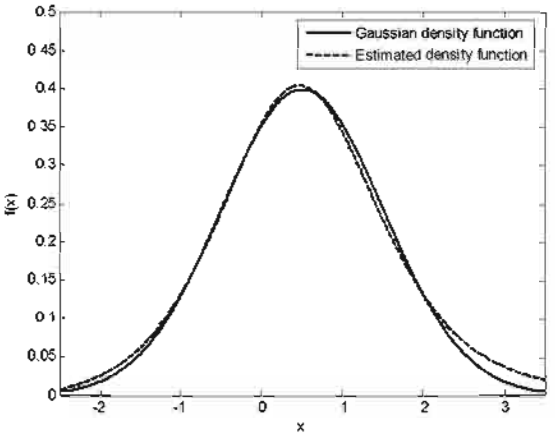

Figure 10: Gaussian density function vs. estimated density function ($\mu = 0.5$ and $\sigma^2 = 1$)

Figure 11 shows the results for the Gaussian distribution function with $\mu = 0.8$ and $\sigma^2 = 4$. It has a MSE of 1.33×10^{-5} for 1000 epochs. The result for the Gaussian density function is shown in Figure 12. It has a MSE value of 2.22×10^{-5}. The values of λ_s and λ_p are taken to be -0.6 and 1 respectively.

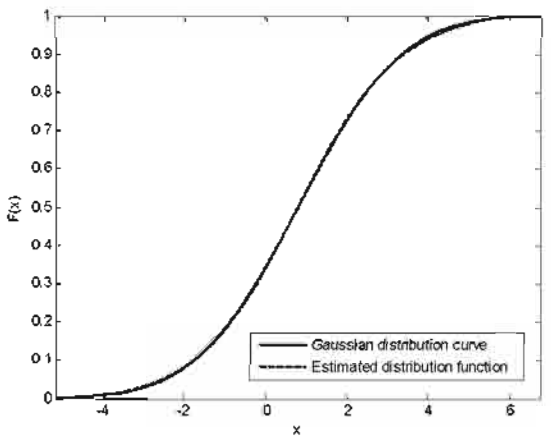

Figure 11: Gaussian distribution function vs. estimated distribution function ($\mu = 0.8$ and $\sigma^2 = 4$)

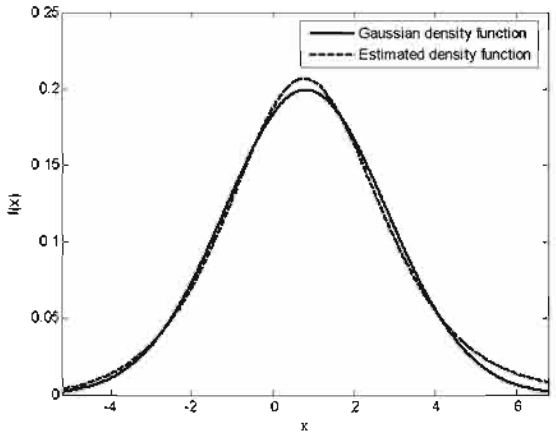

Figure 12: Gaussian density function vs. estimated density function (μ = 0.8 and σ^2 = 4)

The real Gaussian distribution obtained after 1000 epochs for μ = -0.5 and σ^2 = 1is shown in Figure 13. A MSE of 2.22 x 10^{-5} is observed. The real Gaussian density function is shown in Figure 14 and it has an MSE of 1.34 x 10^{-4}. The values of λ_s and λ_p are taken to be 0.9 and 1 respectively.

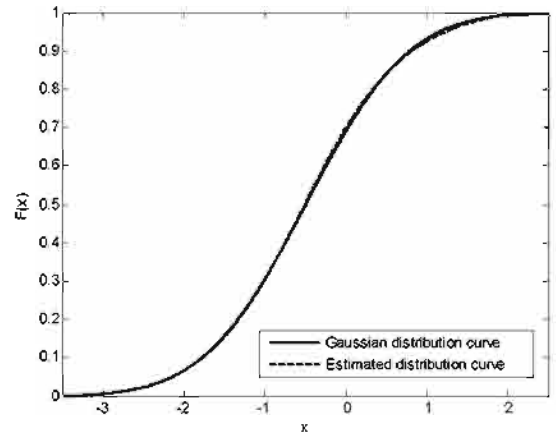

Figure 13: Gaussian distribution function vs. estimated distribution function (μ = -0.5 and σ^2 = 1)

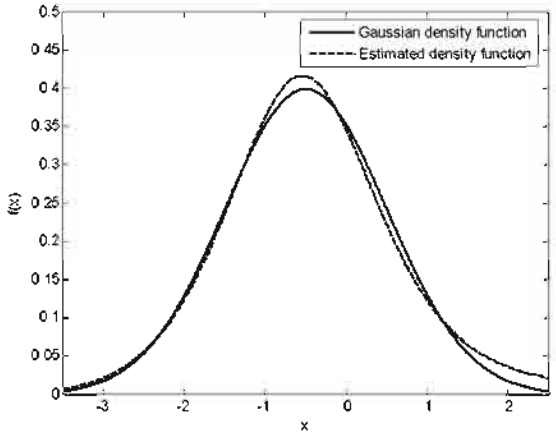

Figure 14: Gaussian density function vs. estimated density function (μ = -0.5 and σ^2 = 1)

It can be seen from the above results that the GN trained with the PSO works well for the estimation of the probability density function for various values of mean and variance.

7 Application for Multi-Sensor Data Fusion

Data fusion is generally used to eliminate the uncertainty and to gain a more precise and reliable value than the arithmetical mean of the measured data from finite sensors. In other words, data fusion seeks to combine data from multiple sensors to perform inferences that may not be possible from a single sensor alone. Multi-sensor data fusion has widespread applications such as automated target recognition, remote sensing, battlefield surveillance, control of autonomous vehicles, monitoring of complex machinery, robotics and medical applications. One of the techniques used for multi-sensor data fusion is to fuse the data using neural networks. The neural networks provide a sub-optimal but practical solution as the training algorithms used for these networks generally converge to the first workable solution as shown earlier [11], [12]. For lesser number of sensors, the conventional neural network can be used, but as the number of sensors increases, the computational time and the memory requirements of the conventional neural network also increases. The GN on the other hand has lesser computational time and also lower memory requirements. Thus, the response would be faster for large number of sensors.

8 Conclusions

A new method for the estimation of the probability density function using the Generalized Neuron has been presented. The training of the GN with the PSO algorithm to approximate distribution functions with different means and variance is shown to be accurate. The differentiation of the GN results in accurate estimates of the respective density functions. The GN can be used to approximate any function provided the characteristic functions f_1 and f_2 (as shown in Figure: 1) are taken according to the function representing the data. The advantage of the GN structure is that it takes fewer parameters than the traditional feedforward neural networks. Thus, the training time required is less and the requirements on the hardware for the implementation are not expensive as for the networks with more parameters. Future work includes applying the GN for the density estimation in multi-sensor data fusion.

References

[1] R. Song, "*Density Estimation Embedded B-spline Smoother for Signal Denoising and Edge-preserving*" Canadian Conference on Electrical and Computer Engineering, Vol. 1, pp. 299 – 302, May 2004.

[2] W. Charytoniuk, M.S. Chen, P. Kotas, P. Van Olinda, "*Demand forecasting in power distribution systems using nonparametric probability density estimation*", IEEE Transactions on Power System, Vol. 14, No. 2, pp. 1200-1206, Nov 1999.

[3] P. Wach, B. Tilg, W. Rucker, R. Stollberger, "Current density estimation within a human heart including a boundary-element-torso model," Engineering in Medicine and Biology Society, Proceedings of the 16th Annual International Conference of the IEEE, Vol. 1, pp. 33-34, Nov 1994.

[4] Dorin Comaniciu, "Density Estimation-based Information Fusion for Multiple Motion Computation." Proceedings of the workshop on Motion and Video Computing, MOTION 02, pp. 241-246, Dec 2002.

[5] J. Llinas, D.L. Hall, "An Introduction to Multi-Sensor Data Fusion." Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, ISCAS '98, vol. 6, pp. 537-540, Jun 1998.

[6] F. Xiongfeng, Y. Xianhui, X. Yongmao, "*A New Method for Density Estimation by Using Forward Neural Network.*" International Joint Conference on Neural Networks 1999, IJCNN'99, vol. 2, pp. 1461-1464, Jul 1999.

[7] D.K. Chaturvedi, O.P. Malik, P.K. Kalra, "*Generalized Neuron based Adaptive Power System Stabilizer.*" IEE Proc.-Gener. Transm. Distrib., Vol. 151, No. 2, Mar 2004.

[8] J. Kennedy and R. Eberhart, "*Particle Swarm Optimization.*" Proceedings, IEEE conference on neural networks, Perth, Australia, vol. 4, pp. 1942-1948, Dec 1995.

[9] Y. Shi and R. Eberhart, "A modified particle swarm optimizer." Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, pp. 69-73, May 1998.

[10] Venu G.Gudise, Ganesh K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks" IEEE, 2003.

[11] Y. Wang, S.D. Goodman, " Data Fusion with Neural Networks." International Conference on Systems, Man and Cybernetics, IEEE 1994, vol. 1, pp. 640-645, Oct 1994.

[12] D.W. Fincher, D.F. Mix, "Multi-sensor Data Fusion Using Neural Networks." IEEE International Conference on Systems, Man and Cybernetics, pp. 835-838, Nov 1990.