CHAPTER 5

SELECTION OF DISK-SEEK ALGORITHM USING FUZZY NEURAL NETWORKS WITH MLM ALGORITHM

5.1 Introduction

The BPN learning method used widely in multilayer neural networks has a possibility of getting trapped in local minima due to inadequate weights and insufficient number of hidden nodes. Therefore an enhanced fuzzy neural network using self organization that self-generates hidden nodes is used. The Fuzzy Neural Networks (FNNs) is trained with Modified Levenberg Marquardt algorithm called Modified LM based fuzzy neural networks (LMFNN) is used for effective selection of the appropriate disk-seek algorithm. The complier will use this disk seek algorithm and the disk access pattern to restructure the source code to reduce the power consumption by inserting clear disk power management calls in suitable place of the source code.

5.2 Overview of Fuzzy Neural Networks

Both neural networks and fuzzy systems have certain common characteristic features. They can be employed for solving issues such as pattern recognition, regression or density estimation if no mathematical model of the given problem exists. They exclusively do have certain drawbacks and benefits which almost entirely evaporate by integrating both concepts.

Neural networks can become a considerable approach only if the issue is uttered by an adequate amount of observed samples. These observations are utilized to train the black box. Alternatively, no prior knowledge about the issue is required. But, it is not easy to extract logical rules from the neural network's structure.
In contrast, a fuzzy system requires linguistic rules rather than learning samples as former knowledge. Moreover, the input and output variables have to be illustrated linguistically. If the knowledge is imperfect, wrong or opposing, then the fuzzy system must be tuned. As there is no formal technique for it, the tuning is carried out in a heuristic technique. This is generally very time consuming and error-prone.

When compared to a general neural network, connection weights and propagation and activation functions of fuzzy neural networks differ a lot. Even though, there are various techniques to model a fuzzy neural network, most of the techniques posses certain characteristic features as listed below:

- A neuro-fuzzy system based on a fundamental fuzzy system is trained through data-driven learning technique obtained from neural network theory. This heuristic only considers local data to cause local alterations in the fundamental fuzzy system.
- It can be denoted as a group of fuzzy rules at any time of the learning process, i.e., before, during and after.

Therefore, the system could be initialized with or without prior knowledge through fuzzy rules.

- The learning process is constrained to assure the semantic attributes of the underlying fuzzy system.
- A neuro-fuzzy system approximates an n-dimensional unknown function which is partially denoted by training samples.

Therefore, Fuzzy rules can be interpreted as indistinct prototypes of the training data.

- A neuro-fuzzy system is denoted as special three-layer feed forward neural network as shown in Figure 5.1.
Figure 5.1 Architecture of a Neuro-Fuzzy System

- The first layer corresponds to the input variables.
- The second layer symbolizes the fuzzy rules.
- The third layer denotes the output variables.
- The fuzzy sets are altered as (fuzzy) connection weights.
- Certain techniques also utilize five layers where the fuzzy sets are encoded in the units of the second and fourth layer, respectively. But, these models can be changed into three-layer architecture.

5.3 Fuzzy Systems

Fuzzy systems recommend a mathematic calculus to interpret the subjective human knowledge of the real processes. This is a technique to influence practical knowledge with certain level of uncertainty. The fuzzy set theory was initiated by Lofti Zadeh (1965). The nature of such systems is illustrated by means of a group of fuzzy rules, like:

IF <premise> THEN <consequent>
that uses linguistics variables with symbolic terms. Each term denotes a fuzzy set. The terms of the input space (typically 5-7 for each linguistic variable) create the fuzzy partition.

Advantages of the fuzzy systems are
- ability to denote inherent uncertainties of the human knowledge with linguistic variables;
- simple interaction of the expert of the domain with the engineer designer of the system;
- easy interpretation of the results, due to the representation of the natural rules;
- easy extension of the base of knowledge via the addition of new rules;
- Robustness in relation to the possible disturbances in the system.

5.4 Neuro Fuzzy Systems

As fuzzy systems are playing a vital role in industrial applications, the community recognized that the growth of a fuzzy system with good performance is very tough. The issue of identifying membership functions and suitable rules is often a very tough mechanism. This results in applying learning algorithms to the fuzzy systems to enhance the performance of the system. The neural networks, that have competent learning algorithms, had been proposed as an option to automate or to support the growth of tuning fuzzy systems. Neural networks and fuzzy systems can be integrated for better performance. Neural networks establishes its computational characteristics of learning in the fuzzy systems and obtain from them the interpretation and clarity of systems representation. Thus, the disadvantages of the fuzzy systems are compensated by the capacities of the neural networks. These techniques are complementary, which justifies its use together.
5.5 FNN Architecture

Here fuzzy neural network is used for selecting the optimal disk seek algorithm among FCFS Planning, SSF Planning and SCAN Planning algorithms which were considered for the experiment. The following inputs were considered for the proposed fuzzy neural network: Xi -> input variables, where X1 -> Starting disk, X2 -> Ending Disk, X3 -> Distance of Disk Head. Y -> disk seek time is the output. The reason for selecting these inputs is to calculate the distance of disk head movement from the current disk head position. This will be used during the computation of disk seek time. The value of these inputs considered in our experiment is the benchmark values of disk parameters.

The proposed fuzzy neural network is a four-layered FNN that is comprised of the input, membership function, fuzzy rules, and output layers as shown in Figure 5.2.

Each node in first layer corresponds to one linguistic label (short, long, etc.) of one of the input variables in Layer 1. Here the linguistic labels/variables represent the distance of disk head movement (short, long, etc) used in computing the disk seek time. In other words, the membership value which specifies the degree to which an input value belongs a fuzzy set.

![Figure 5.2 Structure of the four layered Fuzzy Neural Network](image-url)
Layer 2 is to compute the membership values, whose nodes denote the terms of the relevant linguistic variables. Fuzzy rules are denoted by the nodes at Layer 3. The links before Layer 3 denote the preconditions of fuzzy rules and the links after Layer 3 denote the effects of fuzzy rules.

Layer 4 is the output layer. This four-layered network recognizes the following form of fuzzy rules:

Rule \( R_j \): IF \( x_1 \) is \( A_{1j} \) and ... \( x_i \) is \( A_{ij} \) .... and \( x_M \) is \( A_{Mj} \), THEN \( y \) is \( d_j, j=1, 2, \ldots, N \).

Where \( A_{ij} \) represent the fuzzy sets of the input variables \( x_i, i=1, 2, \ldots, M \) and \( d_j \) are the consequent parameter of \( y \). For the simplicity of analysis, a fuzzy rule 0 is added as

Rule 0: IF \( x_1 \) is \( A_{10} \) and \( x_M \) is \( A_{M0} \), THEN \( y \) is \( d_0 \).

Where \( A_{k0} \) denotes a universal fuzzy set, whose fuzzy degree is 1 for any input value \( x_i, i=1, 2, \ldots, M \) and \( d_0 \) represents the resulting parameter of \( y \) in the fuzzy rule 0. \( O^{(p)} \) and \( a^{(p)} \) are defined as the output and input variables of a node in layer \( P \), respectively.

The signal propagation and the fundamental functions in each layer are illustrated as follows.

Layer 1 – Input layer: No calculation is done in this layer. Each node in this layer, which is equivalent to one input variable, only transmits input values to the next layer directly. That is

\[
O^{(1)} = a_i^{(1)} = x_i
\]

(5.1)

Where \( x_i, i=1, 2, \ldots, M \) are the input variables of the FNN.

The inputs and the associated link weights are given equation 5.1 in page no. 88. The superscript denotes the layer number. The action of each node is to output an activation value as a function of its input. Function \( a \) denotes the activation function. The link weight in layer one is unity. Each node in the first layer, which is
equivalent to one input variable, only transmits input values to the next layer directly.

Layer 2: Membership function layer: Each node in this layer is a membership function that is equivalent to one linguistic label of one of the input variables in Layer 1. On the other hand, the membership value which indicates the degree to which an input value belongs to a fuzzy set is computed in Layer 2:

\[ O^{(2)} = u_i^{(j)}(a_i^{(2)}) \]  \hspace{1cm} (5.2)

Where \( u_i^{(j)} \) is a membership function \( u_i^{(j)}: R \rightarrow [0, 1], \) \( i=1, 2, \ldots, M, \) \( j=1, 2, \ldots, N. \)

With the use of Gaussian membership function, the operation performed in this layer is

\[ O^{(2)} = e^{-\frac{(a_i^{(2)}-m_{ij})^2}{\sigma_{ij}^2}} \]  \hspace{1cm} (5.3)

Where \( m_{ij} \) denotes the center (or mean) and \( \sigma_{ij} \) represents the width (or variance) of the Gaussian membership function of the \( j \)-th term of the \( i \)-th input variable \( x_i. \)

The membership value which indicates the degree to which an input value belongs to a fuzzy set is computed in Layer 2 by using the Gaussian membership function along with the activation function.

Layer 3 – Rule layer: A node in this layer denotes one fuzzy logic rule and carry out precondition matching of a rule. Here, use the following AND operation for each Layer 3 node

\[ O^{(3)} = \prod_{i=1}^{M} a_i^{(3)} \]  \hspace{1cm} (5.4)

\[ = e^{-[D_j(x-m_j)]^T[D_j(x-m_j)]} \]

Where \( D_j = diag\left[\frac{1}{\sigma_{1j}}, \ldots, \frac{1}{\sigma_{Mj}}\right], \) \( m_j = [m_{1j}, m_{2j}, \ldots, m_{Mj}]^T \)

\[ x = [x_1, x_2, x_3, \ldots, x_M]^T. \]
is the FNN input vector of FNN. The output of a Layer-3 node denotes the firing strength of the equivalent fuzzy rule.

Layer 4 – Output layer. The single node $O^{(4)}$ in this layer is labeled with $\beta$, which calculates the overall output and can be computed as:

$$O^{(4)} = \sum_{j=1}^{N} d_j \times a_j^{(4)} + d_0 \quad (5.5)$$

Where the connecting weight $d_j$ is the output action strength of the Layer 4 output associated with the Layer 3 rule and the scalar $d_0$ is a bias. Thus the fuzzy neural network mapping can be rewritten in the following input-output form:

$$O^{(4)} = \sum_{j=1}^{N} d_j \times a_j^{(4)} + d_0 \quad (5.6)$$

$$= \sum_{j=1}^{N} d_j \prod_{i=1}^{M} u_i^{(j)} + d_0$$

5.6 Construction of Fuzzy Rules

Suppose that there are $n$ requests $R_1, ..., R_n$. The processing time $p_i$ are deterministic and the deadlines are $d_i$. But a certain delay is tolerable up to a Late date $d_i^*$ beyond which the request will be canceled, because the customer will resort to other services. As completion time of job $J_i$ passes between Due date $d_i$ and Late date $d_i^*$, the satisfaction of request decreases until it vanishes at the latter. Fig. 5.3 shows a fuzzy deadline and its associated membership function.

The greater the delay, the lower the satisfaction. In this research, preemptions for the requests aren’t admitted and the object is to find optimal assignment of the priorities using classical ordering of fuzzy sets.
Let $C_i$ denote completion time of job $J_i$ and the membership function $S_i(C_i)$ denoted degree of satisfaction is associated with each request and its deadline, which is defined as follows:

$$S_i(C_i) = \begin{cases} 
1 & C_i \leq d_i \\
1 - \frac{(C_i - d_i)}{(d_i^* - d_i)} & d_i < C_i \leq d_i^* \\
0 & d_i^* < C_i
\end{cases} \quad (5.7)$$

If complete time $C_i$ is before due date $d_i$, there is maximum degree of satisfaction that is one. When complete time delays beyond $d_i^*$, level of satisfaction will decrease to zero. Generally, $C_i$ is in the interval $(d_i, d_i^*)$, and $u(C_i)$, degree of satisfaction with respect to $C_i$, belongs to $(0, 1)$. Let the feasible schedule is denoted by $\Pi$. Then minimum degree of satisfaction with schedule $\Pi$ is

$$S_{\text{min}} = \min \left\{ S_i(C_i(\Pi)) \mid i = 1, 2, \ldots, n \right\} \quad (5.8)$$

Under the above setting, consider the following problem $P$:

Maximize $S_{\text{min}} = \min \left\{ S_i(C_i(\Pi)) \mid i = 1, 2, \ldots, n \right\}$

Subject to $S_i(C_i(\Pi)) > 0 \quad (5.9)$
That is to get an optimal schedule $\Pi$ which maximizes the minimum satisfaction with respect to fuzzy deadlines of jobs. Now let $t = s_{\text{min}}$. Obviously $0 \leq t \leq 1$. Then from formula (5.7) the inequalities
\[ C_i \leq d_i(1-t) e_i \text{ and } e_i = d_i^* - d_i \text{ for } 1 \leq i \leq n \]
(5.10)
must hold. Next, employ the five rules to solve above problem $P$.

**Rule 1.** If $t > 0$, then each optimal disk schedule satisfies inequality.

**Rule 2.** If $t = 0$, then, for each disk schedule, there exists request $R_k$ such that
\[ d_i(1-t) e_k < C_k \]
(5.11)

For each $0 < t < 1$, let $D_{di}(t)$ be modified deadlines for completion of requests:
\[ D_i(t) = d_i(1-t) e_i \text{ for } 1 \leq i \leq n \]
(5.12)

**Rule 3.** Only those disk schedules are feasible, whose modified deadlines $D_i(t)$ $(1 < i \leq n)$ satisfy following inequality for some $t (0 < t < 1)$:
\[ C_i \leq D_i(t) \]
(5.13)

If no such a feasible schedule exits, then every disk schedule is optimal and the maximum of the minimum degree of satisfaction is zero.

**Rule 4.** For each fixed value of $t$, the optimal assignment of priorities exists if and only if the priorities induced by ordering the requests in the non-decreasing order of $D_i(t)$ are feasible.

Since the modified deadlines change accordingly with $t$, determine the values as:
\[ t_{ij} = 1 + \frac{d_i - d_j}{e_i - e_j} \text{ for some } i, j, i \neq j \]
(5.14)

Rank all the quantities $t_{ij} (0 < t_{ij} < 1)$ in increasing order and create the interval $[t_k, t_{k+1}] (k=0, \ldots, k_{\text{max}})$. All of these intervals follow such rule.
Rule 5. For any \( t \) in the interval \([t_k, t_{k+1}]\), the requests must have same priorities. If \( t \) passes from interval \([t_k, t_{k+1}]\) to interval \([t_{k+1}, t_{k+2}]\), the requests must change the priorities.

Above rules suggest the following procedure for finding an optimal disk schedule. First, find all points of intersection \( t_\theta \) of \( D_i \) and \( D_j \) satisfying \( 0 < t < n \), then sort them and employ a binary search for finding the maximum among them for which a feasible schedule exits according rule 4. If no feasible schedule exits, then every disk schedule is optimal. The algorithm returns the resulting schedule \( \prod \) as a permutation, i.e., \( \prod \) returns the index \( \prod(i) \) of the request in the \( \theta \)th for each \( 1 \leq i \leq n \).

Algorithm

Step 1. Find all \( 0 \leq t \leq 1 \) such that \( t = \frac{e_i - e_j + d_i - d_j}{e_i - e_j} \), for some \( i \neq j \). Let \( \pi \) be an optimal schedule.

Step 2. Let \( T \) be a set of such \( t \). Add 1 to \( T \). Sort all items in \( T \) in decreasing order such that \( t=1 \) is first one. Choose the initial \( t \) from \( T \).

Step 3. Find schedule \( S \) according to the non-decreasing order of \( D_i(t) \). If \( C_i(s) \leq D_i(t) \) for all \( 1 \leq i \leq n \), then \( \pi \leftarrow S \), stop; else go to step 4.

Step 4. Select next \( t \) from \( T \). Go to step 3.

5.7 Experimental Results

The evaluation results of the proposed approach are provided in this section. The energy saving comparison for various power loads are evaluated and compared.

Figure 5.3 shows the energy saving of the proposed LMFNN approach. It is observed from the graph that for different work load applied to the LMFNN power management technique, the energy saving obtained is significant. For example, for \(<\text{exp}, 100>\), the energy saving obtained is 68 %. Similarly for \(<\text{exp}, 500>\) and \(<\text{exp}, 1000>\), the energy saving obtained is 76 and 83 respectively. For \(<\text{par}, 50>\) and \(<\text{par}, 100>\) the energy saving obtained is 74% and 78% respectively.
Figure 5.4: Comparison of Energy Savings for Exponential and Pareto Methods

Figure 5.5: Impact of Stripe Size on Energy Consumption

The normalized energy consumptions with the different stripe sizes are shown in Figure 5.4. It is obvious that the energy savings brought by the LMFNN approach increase as the stripe size increases. There is not much of idleness when the stripe size is very small (16 Kbyte). Actually, the disk idleness in this case becomes so little that even code restructuring cannot take much gain of it. Additional disk requests can be serviced by a single stripe. When the stripe size is increased, which means that, the stripe-level data reuse also improves. It is observed that the proposed LMFNN disk management scheme generates the best savings of about 0.69% with 256 Kbyte stripe size.
Figure 5.6: Impact of Stripe Factor on Energy Consumption

The normalized energy values under various stripe factors shown in Figure 5.5. Figure clearly depicts that when the number of disks is increased, disk idleness increases, and as a result, the compiler approaches show a better behavior. For instance, when the number of disks is very high, 32, the disk idleness attains a very high level. From the Figure 5.5, proposed LMFNN Disk Management scheme is observed to obtain less energy consumption for all the stripe factors.

Figure 5.7 Impact of Starting Disk on Energy Consumption
Figure 5.6 indicates the impact of starting disk on the energy consumption. In order to perform this set of experiments, a random integer number is constructed (for each array in the mgrid benchmark) between 1 and 8 to choose the disk from which the array is striped. Figure 5.6 shows the results for five such experiments. It is observed from the graph that the proposed LMFNN disk management scheme consumes less energy.

The normalized energy consumption values given in Figure 5.5, 5.6 and 5.7 are the normalized values with respect to the system that does not employ any power management strategy.

Figure 5.5, 5.6 and 5.7 shows the impact of the stripe size, stripe factor and starting disk on energy consumption. It is observed that the proposed LMFNN Disk Management approach has good and fair energy consumption in all the three cases.

Thus, the proposed LMFNN Disk Management approach out performs the other three approaches in terms of energy consumption.

5.9 Summary

This chapter shows the software based approach for disk power management by utilizing Fuzzy Neural Network trained with modified levenberg-marquardt algorithm for selecting appropriate disk-seek algorithm. The disk layout information of disk resident data and the data access pattern are used to construct the disk access pattern. The compiler uses the disk access pattern and the selected disk seek algorithm to restructure the application code to reduce the power consumption, by inserting clear disk power management calls in suitable place of the source code. The experimental results show that LMFNN power management approach reduces energy consumption.