CHAPTER 7

Enhanced Kernel Clustering Based
Modular PNN Ensemble Classifier for
Large Dataset PD Signature Patterns*  

* Partial contents of this Chapter has been published in


CHAPTER 7

Enhanced Kernel Clustering Based Modular PNN Ensemble Classifier
for Large Dataset PD Signature Patterns

7.1.1 Kernel based Clustering and Expectation Maximization with Maximum Likelihood (EM-ML)

Expectation Maximization (EM) algorithm is a two-step statistical iterative procedure that utilizes the expectation process (E-step) accompanied by the maximization process (M-step) to obtain the Maximum Likelihood (ML) estimates for large feature datasets that are invariably accompanied by noise and missing points. It is important to note that the EM algorithm is guaranteed to converge to an ML estimate [133, 134] at a reasonably fast computational rate. Each E-step computes the expected value of a set of ‘unobserved’ data by means of the recent parameter estimate and the observed data. Each M-step uses the value obtained from the E-step supposing it to be the measured data for obtaining a likelihood function and in turn in determining an ML estimate of the parameter. In both the steps an iterative procedure is pursued until the change in the log posterior likelihood function is minimized.

From the perspective of training the OPNN version, it is pertinent to note that in each E-step and M-step, the mean is constantly tuned until the log posterior likelihood function is minimized. Since the focus is on obtaining the estimates for the mean of Gaussian distribution of the training datasets, the EM algorithm utilizes a weight parameter, which is also updated during the computation of each step. The E step uses the PDF estimated in the second layer of the PNN along with the intra-class mixing coefficient (β) to estimate
the weight parameter. The M-step uses the data estimated in the E-step and the weight parameter $w_{E,i}$ to form a likelihood function and serves in acquiring the ML estimate. Hence, using the weight calculated from the E step, the new values for the cluster center ($c_{E,i}$) and the intra-class mixing coefficients ($\beta_{E,i}$) are computed.

Researchers who have applied the HRPNN using the EM-ML scheme in a variety of applications, including that which has been utilized in this research study, have encountered difficulties in obtaining the optimum estimates which may be invariably traced to the value of the variance parameter approaching zero when the data are sparsely distributed. The issue pertaining to such numerical difficulties are attempted to be resolved by utilizing a statistical tool for better interval estimation called the Jack-knife method. Figure 7.1 illustrates the flow chart of the generic ML-EM algorithm for implementation with HRPNN.
**Figure 7.1:** Flow Chart of ML-EM Algorithm

**Input Data:**
1. Number of Classes (Q)
2. Dimensionality of Input Data (d)
3. Number of Patterns of Class (B)
4. Number of Training Samples (M)
5. Input for Testing (x)

**ML with EM Algorithm:**
Obtain the Maximum Likelihood (ML) Estimate of the Weights, Centers and Variance of each Class (using Lagrangier Multiplier Operator):

1. **Weight (w):**
   \[
   w_{B,i}^{(Q)}(x_{n,i}) = \frac{\beta_{B,i}^{(Q)} p_{B,i}^{(Q)}(x_{n,i})}{\sum_{m=1}^{B_i} \beta_{m,i}^{(Q)} p_{m,i}^{(Q)}(x_{n,i})}
   \]

2. **Center (c):**
   \[
   c_{B,i}^{(R+1)} = \frac{\sum_{n=1}^{N_i} w_{B,i}^{(Q)}(x_{n,i}) x_{n,i}}{\sum_{n=1}^{N_i} w_{B,i}^{(Q)}(x_{n,i})}
   \]

3. **Mixing Coefficient (β):**
   \[
   \beta_{B,i}^{(Q+1)} = \frac{1}{N_i} \sum_{n=1}^{N_i} w_{B,i}^{(Q)}(x_{n,i})
   \]

**Compute Probability Density Function Estimate:**

\[
p_{B,i}^{(Q)}(x_{n,i}) = \frac{1}{(2\pi \sigma_i^{(Q)})^{d/2}} \exp \left( -\frac{\|x_{n,i} - c_{B,i}^{(Q)}\|_2^2}{2\sigma_i^{(Q)}} \right)
\]

**Class Conditional Probability Density Function Estimate:**

\[
f_{j}(x) = \sum_{i=1}^{B} \beta_{i,j} p_{i,j}(x) \quad 1 \leq j \leq Q
\]

**Bayes Classifier Decision:**

\[
O_{Bayes}(x) = \underset{1 \leq j \leq Q}{\arg \max} \left\{ \alpha_{j} f_{j}(x) \right\}
\]
7.1.2 Homoscedastic and Heteroscedastic PNN with EM-ML Algorithm for PD Pattern Discrimination

Though it is evident from the architecture of the OPNN that more the training patterns the better is the classification accuracy, it is also imperative to investigate the effect of over training and its impact on classification capability more-so, in the context of large dataset training exhibiting high degree of inseparability of decision boundaries. Hence, it would only be appropriate and rational that a smaller set of kernel functions that are representatives of the training data provide a meaningful representation without compromising the accuracy of classification. Streit and Luginbuhl [92] implemented the Expectation Maximization (EM) algorithm to obtain an optimal Maximum Likelihood (ML) training algorithm for mixture of Gaussian PNN with common covariance matrices i.e. common variance parameter with the “same scatter” which is termed as Homoscedastic PNN (HOPNN). This also implies that if the Gaussian functions are provided with the capability to obtain different variances, an even better set of adequate yet more parsimonious set of centers that describe the class conditional PDFs could be utilized. Thus, utilizing uncorrelated Gaussian kernel functions with different variance (“different scatter”) is called the Heteroscedastic PNN (HRPNN). This aspect is shown indicatively in Figure 7.2.
The architecture of HRPNN (a modified supervised version of PNN) [93] is fundamentally the same as the original PNN with a few major changes in the exemplar layer (second layer) and the way in which the winner is selected in the output layer during training. HRPNN is a four-layer feed forward neural network with the first layer, which accepts input patterns. The nodes in the second layer are divided into \( k \) groups, one for each class. The generic second layer node, the \( i^{th} \) kernel in the \( j^{th} \) group, is defined as a Gaussian basis function

\[
p_{i,j}(x) = \frac{1}{(2\pi\sigma_{i,j}^2)^{d/2}} e^{-\frac{\|x-c_{i,j}\|^2}{2\sigma_{i,j}^2}}
\]

where \( c_{i,j} \) is the centre or the mean vector and \( \sigma_{i,j}^2 \) is the positive variance or smoothening parameter.
Each node in the second layer is also called a ‘pattern unit’. The third layer has \( k \) nodes

\[
f_j(x) = \sum_{i=1}^{R_{i,j}} \beta_{i,j} P_{i,j}(x), 1 \leq j \leq R
\]

and each node estimates a class conditional PDF \( f_j \) using a mixture of Gaussian kernels, where \( R_{i,j} \) refers to the number of pattern units for class \( j \) and \( \beta_{i,j} \) is the positive mixing coefficient satisfying the condition \( \sum \beta_{i,j} = 1, 1 \leq j \leq R \).

The fourth layer of the HRPNN makes the decision according to the equation

\[
g(x) = \arg \max \{ \alpha_j f_j(x) \}
\]

where \( \alpha_j \) refers to the class a-priori probability. The architecture of the HRPNN is shown in Figure 7.3.
Figure 7.3: Architecture of Heteroscedastic Probabilistic Neural Network
7.2 Robust Heteroscedastic PNN

The issue pertaining to such numerical difficulties are attempted to be resolved by utilizing a powerful statistical tool for better interval estimation called the Jack-knife method. The Jack-knife procedure partitions a sample space into subsets and observes the influence of each subset during the estimation process. Another major characteristic of the Jack-knife method is that it can remove the effect of outlier in the dataset hyperspace as deliberated in [135]. The modifications of the ML-EM algorithm so as to include the Jack-knife procedure in addition to choosing the starting values of the Gaussian kernel centers as indicated in [135] enables the implementation of the RHRPNN which is summarized in Figure 7.4.
Figure 7.4: Flow Chart of Training Algorithm for Robust Heteroscedastic Probabilistic Neural Network Using EM-ML with Jackknife Procedure
7.2.1 Performance of HOPNN, HRPNN and RHRPNN for Large Dataset PD Signature Classification

1. The classification rate of HRPNN is about 1-2% better (about 10 to 15 lesser number of misclassified patterns) thus indicating the inherent capability of this version of PNN in creating independent class density estimates in addition to separate variance related to each class which proves to be vital in improving the classification performance of HRPNN version.

2. It is also evident from Table 7.1 and based on the analysis reported in [136] that a more parsimonious set of centers is obtained utilizing the HRPNN algorithm with superior classification capability. However, problems related to the effect of outliers in discriminating inter-class clusters is observed to pose some difficulties during classification of patterns pertaining to relatively fully overlapped patterns.

3. Since rapid computation speed and convergence is observed with both the proposed PNN versions (for all the feature vectors) this methodology provides an exciting opportunity for creating a framework for implementing a real-time condition monitoring system for classifying divergent categories of insulation flaws in electrical power apparatus.
Table 7.1: Classification Capability of HOPNN and HRPNN with EM-ML Algorithms

<table>
<thead>
<tr>
<th>Input Feature</th>
<th>No. of Tuples</th>
<th>Total No. of Testing Datasets</th>
<th>Randomly Chosen Initial Centers-144 Nos. (36 Sets for each class)</th>
<th>Classification Capability EM-ML Algorithm (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No. of PDF labeled sequentially as EC, AC, OC and ECC (Set 1 only)</td>
<td>HOPNN</td>
</tr>
<tr>
<td>φ=q_{max}-n (30°)</td>
<td>36</td>
<td>621</td>
<td>24,22, 20,21</td>
<td>20,18, 18,20</td>
</tr>
<tr>
<td>φ=q_{min}-n (30°)</td>
<td>36</td>
<td>621</td>
<td>22,25, 24,18</td>
<td>18,17, 16,18</td>
</tr>
<tr>
<td>φ=q_{max}-n (10°)</td>
<td>108</td>
<td>624</td>
<td>12,14, 15, 14</td>
<td>10,12, 13, 9</td>
</tr>
<tr>
<td>φ=q_{min}-n (10°)</td>
<td>108</td>
<td>624</td>
<td>16,11, 16,12</td>
<td>13,11, 9, 8</td>
</tr>
<tr>
<td>Traditional Statistical Operators (30°)</td>
<td>48</td>
<td>621</td>
<td>14,16, 14,10</td>
<td>12,15, 11,10</td>
</tr>
<tr>
<td>Traditional Statistical Operators (10°)</td>
<td>144</td>
<td>621</td>
<td>12,14, 18,11</td>
<td>14,11, 15,13</td>
</tr>
</tbody>
</table>

7.2.2 Analysis of the performance of RHRPNN

1. It is obvious from studies [136] that RHRPNN results in more parsimonious set of representative centres (PDF) for training the PNN in addition to rapid convergence towards the optimum set of centres without compromising the classification capability. This is also summarised in Table 7.2.

2. It is evident that a minor change in the number of PDFs leads to a larger misclassification rate with a subsequent impact on the requisite number of iteration for an optimal convergence during training. It is also observed that the RHRPNN yields an optimal set of PDFs similar to that obtained with the HRPNN yet being able to counter the influence of outlier exceptionally well. The superiority of the robust procedure in
ensuring a much more parsimonious set of centres in addition to addressing issues pertaining to outliers is also made evident from Table 7.3.

**Table 7.2:** Capability of HRPNN and Robust HRPNN in Multi-source PD Pattern Classification

<table>
<thead>
<tr>
<th>Input Feature Vector</th>
<th>PD Types</th>
<th>Total No. of Training Datasets</th>
<th>Testing Datasets</th>
<th>No. of Centers (PDFs)</th>
<th>No. of Iterations</th>
<th>No. of Misclassifications</th>
<th>Classification Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single Source</td>
<td>3 (90 sets/ type)</td>
<td>120</td>
<td>22,20,21</td>
<td>26</td>
<td>8</td>
<td>93.33%</td>
</tr>
<tr>
<td></td>
<td>Multi-Source: Partially overlapped</td>
<td>4 (120 sets/ type)</td>
<td>160</td>
<td>24,22,20,21</td>
<td>28</td>
<td>12</td>
<td>92.5%</td>
</tr>
<tr>
<td></td>
<td>Single Source</td>
<td>3 (90 sets/ type)</td>
<td>120</td>
<td>24,21,18</td>
<td>24</td>
<td>10</td>
<td>91.67%</td>
</tr>
<tr>
<td></td>
<td>Multi-Source: Partially overlapped</td>
<td>4 (120 sets/ type)</td>
<td>160</td>
<td>20,25,24,28</td>
<td>30</td>
<td>10</td>
<td>93.75%</td>
</tr>
<tr>
<td></td>
<td>Single Source</td>
<td>3 (90 sets/ type)</td>
<td>120</td>
<td>20,25,38</td>
<td>23</td>
<td>5</td>
<td>95.83%</td>
</tr>
<tr>
<td></td>
<td>Multi-Source: Partially overlapped</td>
<td>4 (120 sets/ type)</td>
<td>160</td>
<td>10,13,13,13</td>
<td>36</td>
<td>7</td>
<td>94.17%</td>
</tr>
<tr>
<td></td>
<td>Single Source</td>
<td>3 (90 sets/ type)</td>
<td>120</td>
<td>15,12,14</td>
<td>12</td>
<td>5</td>
<td>95.83%</td>
</tr>
<tr>
<td></td>
<td>Multi-Source: Partially overlapped</td>
<td>4 (120 sets/ type)</td>
<td>160</td>
<td>10,16,14,20</td>
<td>8</td>
<td>5</td>
<td>97.5%</td>
</tr>
<tr>
<td></td>
<td>Single Source</td>
<td>3 (90 sets/ type)</td>
<td>120</td>
<td>8,14,12</td>
<td>10</td>
<td>3</td>
<td>97.5%</td>
</tr>
<tr>
<td></td>
<td>Multi-Source: Partially overlapped</td>
<td>4 (120 sets/ type)</td>
<td>160</td>
<td>12,14,13,11</td>
<td>7</td>
<td>6</td>
<td>98.23%</td>
</tr>
</tbody>
</table>

**Inferences on the Performance of HRPNN and RHRPNN Modular Versions in Classifying Large Dataset PD Signatures**

It is evident from this research that the innovative approach of using HRPNN and RHRPNN for multiple source PD pattern classification task has the following distinct advantages:
1. Once the training phase of the HRPNN is well established, lesser number of exemplars may be used to describe the probability density estimate of that class. Further, fast convergence of the iterative procedure adopted by the supervised training algorithm is also perceptible though not usually the case with a few other types of supervised ANNs.

2. An even more frugal and optimum set of centres pertaining to each class of PD source is obtained in the case of RHRPNN than the HRPNN with both types of pre-processing schemes. Table 7.3 establishes this aspect clearly.

3. The RHRPNN in addition to providing an optimal set of centres addresses substantially the effect of outliers during the training phase. This is made possible by the Jack-knife estimation procedure which aids in carrying out the hold-one-out validation of the input dataset at a rapid computational speed thus reducing the computational cost during large dataset training. This aspect is also deliberated in Section 4.3.

4. As discussed in Section 4.3 and later summarised subsequently in the same section, though the HRPNN and the RHRPNN do not yield exceptional classification rate with regard to varying applied voltages, it is evident that both tools provide a frugal set of centres for one distinct applied voltage dataset training that are classified excellently. This provides a viable tool for developing a hybrid scheme as a part of future research.

5. The adaptation of the TPSW scheme used by researchers of allied areas of engineering to that of PD dataset as a pre-processing and feature extraction technique is observed to have considerable success which is made evident during the detailed analysis in Section 4.2 and Section 4.3 and summarised in Table 7.2. It is hence perceived that the substantial success of the TPSW scheme will provide an exciting opportunity for further investigations while utilizing other variants of the PNNs during future research studies.
7.3.1 Orthogonal Least Square Center Selection Algorithm

Orthogonal Least Square (OLS) is an efficient technique for subset model representation and selection [91]. In contrast to many NN-learning algorithms, which necessitate a predetermined network structure, this algorithm computes and resolves the appropriate basis function network structure during learning. An orthogonal least square scheme is employed as a subset model to identify appropriate basis function centers from the network training data and to estimate the network weights simultaneously in an efficient manner. Notwithstanding, the batch OLS method also chooses the number and the position of the RBF centers. The procedure involves utilizing the Gram Schmidt Orthogonalization (GSO) implemented in two modes namely the forward and the backward center-selection methods. The forward selection approach involves building up a network by adding, one at a time, centers at the data points until an adequate network is created that in turn would result in the largest decrease in the network output error at each stage. On the other hand, the backward selection algorithm is an alternative approach that sequentially removes from the network, one at a time, those centers that cause the smallest increase in the residual. All the training examples are considered as candidates for the centers, and the one that reduces the mean square error the most is selected as a new hidden unit. The GSO scheme is first used to construct a set of orthogonal vectors in the space spanned by the vectors of the hidden activation unit and a new center is then selected by minimizing the residual mean square error. Either of the approaches i.e. Forward OLS (FOLS) or Backward OLS (BOLS) can be utilized for center selection since they have similar computational cost with the advantage
being the exemption of computation of the inversion of matrices which would enable correspondingly quick recursive solutions.

**FOLS Algorithm for Center Selection and Clustering**

The batch FOLS method not only determines the weights but also chooses the number and position of the RBPNN centers. The RBPNN considers a linear regression model wherein two mappings are considered:

\[
D = \Phi V + E \quad \text{(1)}
\]

\[
D = W G + E \quad \text{(2)}
\]

where \(D\) is the desired output matrix, \(\Phi\) is regression matrix, \(V\) is the weight matrix and \(E\) is the error matrix corresponding to the mapping. The objective of the FOLS procedure is to determine the optimal values of matrix \(\Phi\) and \(V\) by minimizing the error matrix \(E\) in order to guarantee convergence and accuracy. In the OLS method the regression matrix \(\Phi\) is decomposed by performing Q–R decomposition. \(\Phi\) corresponds to first mapping called regression matrix represented by

\[
\Phi_{N \times n_h} = [\Phi_1 \Phi_2 \ldots \Phi \ell \ldots \Phi_{n_h}]
\]

\[
\Phi_{N \times n_h} = [\Phi_1 \quad \Phi_2 \quad \ldots \quad \Phi \ell \quad \ldots \quad \Phi_{(N)}]^T \quad \text{(3)}
\]

FOLS method comprises decomposition of matrix \(\Phi\) into

\[
\Phi = W \cdot A \quad \text{(4)}
\]

where \(A\) is an \(n_h \times n_h\) upper triangular matrix.
and $W$ is an orthogonal matrix $W_{(N \times nh)} = (w_1 \ w_2 \ \ldots \ w_{\ell} \ \ldots \ w_{nh})$

$$W^T \cdot W = H \quad \text{------------------- (6)}$$

Hence, $H_{(nh \times nh)}$ is a diagonal matrix.

Also $w_{\ell}^T \cdot w_{\ell} = \sum_{i=1}^{N} w_{\ell}(t)_i w_{\ell}(t)_i = h_{\ell} \ ; \quad 1 \leq \ell \leq n_h \quad \text{--------- (7)}$

$$\quad w_{\ell}^T \cdot w_{j} = 0; \quad (\ell \neq j)$$

Thus substituting (4) in (1) we have

$$A \cdot V = G \quad \text{------------------------ (7)}$$

Also (1) can be written as $D = WG + E \quad \text{------------------------ (8)}$

(i.e.) $d = W \cdot g_m + \varepsilon_m \quad 1 \leq m \leq n_0$

Thus the ideal OLS solution for $G$ is given by

$$\hat{G} = H^{-1} \ W^T \cdot D$$

i.e., $\hat{g}_{lm} = w_{\ell}^T \cdot d_m / (w_{\ell}^T \cdot w_{\ell}) \ ; \quad 1 \leq \ell \leq n_b$

$$1 \leq m \leq n_0$$
A training algorithm based on the mathematical aspects discussed herein is implemented in MATLAB 7.1, Release 14. A flow chart illustrating the various stages of the FOLS training algorithm is presented in Figure 7.5.
Collect ‘N’ Training Set

At the First Step:

Orthogonalize Vector and Obtain $g_{lm}$:
1. Set $n_0 = 1$; (2) for each center $u_1^l = x(l), l = 1, \ldots, N$ find
   \[
   \varphi_1^{(l)} = \left[ \varphi_1^{(l)}(x(1)) \ldots \varphi_1^{(l)}(x(t)) \ldots \varphi_1^{(l)}(x(N)) \right]^T, \quad (l = 1, \ldots, N)
   \]
2. Calculate: $w_1^{(l)} = \varphi_1^{(l)}$
3. Compute Least Square Solution:
   \[
   g_{1m}^{(l)} = \left( w_1^{(l)} \right)^T d_m / \left[ \left( w_1^{(l)} \right)^T w_1^{(l)} \right], 1 \leq m \leq n_0
   \]
4. Compute Reduction in Error:
   \[
   [err\_red]_{1}^{(l)} = \sum_{m=1}^{n_0} g_{1m}^{(l)} \sum_{m=1}^{n_0} \left( d_m d_m \right)
   \]

Select the Error Regressor with the highest reduction in error?

Highest reduction in error?

At $k_{th}$ Step
\[
x(r) \left( 1 \leq r \leq N, r \notin \{l_1, l_2, \ldots, l_{k-1}\} \right)
\]

B

Yes

No

A

If $r < N$

1. Compute: $\varphi_{0r} = [\varphi_{0r}(x(1)) \ldots \varphi_{0r}(x(t)) \ldots \varphi_{0r}(x(N))]^T; \quad (1 \leq r \leq N), r \notin \{l_1, l_2, \ldots, l_{k-1}\}$
2. Calculate:
   \[
   \alpha_{qk}^{(r)} = w_q^T \varphi_k^{(r)} / \left( w_q^T w_q \right), 1 \leq q \leq k
   \]
3. Calculate:
   \[
   w_k^{(r)} = \varphi_k^{(r)} - \sum_{q=1}^{k-1} \alpha_{qk}^{(r)} w_q
   \]
4. Compute:
   \[
   g_{km}^{(r)} = \left( w_k^{(r)} \right)^T d_m / \left[ \left( w_k^{(r)} \right)^T w_k^{(r)} \right], 1 \leq m \leq n_0
   \]

Compute Reduction in Error:
\[
[err\_red]_{k}^{(r)} = \sum_{m=1}^{n_0} g_{km}^{(r)} \sum_{m=1}^{n_0} \left( d_m d_m \right)
\]
Figure 7.5: Flow Chart of Forward Orthogonal Least Square Center Selection Algorithm

7.3.2 Radial Basis Probabilistic Neural Network with Orthogonal Least Square Algorithm- Optimal Center Selection Strategy of PD Signatures during Training

RBPNN [137, 138] comprises four layers viz., input, two hidden and output layer. The first hidden layer consists of the hidden centers determined by input training sample
dataset. The second hidden layer selectively sums the outputs of the first hidden layer and is usually of the same size as that of the output layer. The first hidden layer satisfies the transfer function requirement based on the Parzen window function $Y_i = k(x - c)^2/\sigma^2$ where ‘$\sigma$’ is the smoothing parameter. The second hidden layer is similar to the class layer of the PNN. The training for the weight vector in the third layer (second hidden layer) is equivalent to the output layer of RBFNN. It is evident that the RBPNN paradigm integrates the advantages of the Radial Basis Function Neural Network (RBFNN) and the PNN thus avoids a large number of hidden units of the PNN and incidentally reduces the training time of the RBFNN. It is also clear from the topology of this NN that the selection of the number of neurons in the first hidden layer of the RBPNN is the most significant aspect that influences the improvement of the performance of the neural network. Fewer hidden centers in the first layer improve the training computational speed though with lower convergence accuracy. It is evident that a trade–off between accuracy and computational cost is the key to the selection of optimal number of centers in the first hidden layer. Figure 7.6 shows the architecture of the RBPNN.
7.3.3 Analysis of the performance of RBPNN with OLS Center Selection Algorithm

1. It is pertinent to note that the OLS algorithm is able to create optimal number of centers depending on the input pre-processing scheme. It is observed as reported in [139] that increasing the number of patterns manually for better training in fact leads to less than optimal classification solution during the training phase, concurred by researchers in NN literature. It is also to be noted that the centers are now weight vectors which are representatives of the samples and not the input pattern itself.

2. For lesser number of centers obtained from OLS algorithm, the RBPNN is able to classify the various defect types (both single and multiple) to a considerable degree of success not withstanding the changing applied voltages that lead to considerable changes in the patterns. It is worth noting that the computational time during the training phase is

![Figure 7.6: Radial Basis Probabilistic Neural Network Architecture](image)
also considerably less, though the training involved a noticeably large data size with large dimensionality. These observations indicated in Table 7.3 elucidate the capability of the OLS-RBPNN in classifying PD patterns successfully for large training dataset in addition to complexities due to changes in PD pattern owing to varying applied voltages.

**Table 7.3: Capability of RBPNN in Multi-source PD Pattern Classification**

<table>
<thead>
<tr>
<th>Type of Preprocessing Scheme</th>
<th>Sequence in Phase Window</th>
<th>Total No. of Patterns presented during Training</th>
<th>No. of Patterns as FOLS Centers</th>
<th>Classification Capability (Total No. of Patterns: 400)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measures Based on Maximum Values</td>
<td>( \phi_{\text{max}} ) (30°)</td>
<td>180</td>
<td>112</td>
<td>81%</td>
</tr>
<tr>
<td></td>
<td>( \phi_{\text{max}} ) (10°)</td>
<td>180</td>
<td>94</td>
<td>83%</td>
</tr>
<tr>
<td>Measures Based on Minimum Values</td>
<td>( \phi_{\text{min}} ) (30°)</td>
<td>180</td>
<td>126</td>
<td>80%</td>
</tr>
<tr>
<td></td>
<td>( \phi_{\text{min}} ) (10°)</td>
<td>180</td>
<td>108</td>
<td>82%</td>
</tr>
<tr>
<td>Measures Based on Central Tendency</td>
<td>( \phi_{\text{q-n}} ) (30°)</td>
<td>180</td>
<td>104</td>
<td>85%</td>
</tr>
<tr>
<td></td>
<td>( \phi_{\text{q-n}} ) (10°)</td>
<td>180</td>
<td>72</td>
<td>92%</td>
</tr>
<tr>
<td>Measures Based on Dispersion</td>
<td>( \phi_{\text{q-n}} ) (30°)</td>
<td>180</td>
<td>114</td>
<td>83%</td>
</tr>
<tr>
<td></td>
<td>( \phi_{\text{q-n}} ) (10°)</td>
<td>180</td>
<td>88</td>
<td>89%</td>
</tr>
</tbody>
</table>

**Comparison of Classification Capability of FOLS-RBPNN with Original Version of PNN**

1. The role played by the variance/smoothing parameter during classification has been compared for its performance in PNN and the FOLS-RBPNN. It is obvious that the original version of PNN utilizes a global variance parameter for all classes and hence suffers during generalization and classification while unique centers during the training phase enables in better class-separability in the case of RBPNN.
2. Furthermore, one among the variant versions of PNN namely the HRPNN was also taken up for comparison of the performance of the proposed FOLS-RBPNN. The HRPNN, a supervised version, utilizes an Expectation Maximization (EM) with Maximum Likelihood (ML) algorithm to provide parsimonious centers (Probability Density Functions ‘PDF’) pertaining to each class. It is observed during detailed studies that though a parsimonious set of centers are obtained during the training phase of the HRPNN, only patterns presented to the NN for training are correctly classified while several patterns are misclassified during the testing phase (the classification capability is of the order of 65-70%). This aspect may be attributed to obvious deficiencies of the EM and ML algorithm to handle large dimensionality dataset and its tendency during training to get trapped to local minima.

3. It is also observed that selectively chosen centers leads to better classification while in most cases it results in wide misclassifications. It is of significant interest to note that during the training phase the patterns are reasonably classified while large numbers of misclassifications are observed during the testing phase. Table 7.4 summarizes this aspect.

<table>
<thead>
<tr>
<th>Type of Preprocessing Scheme</th>
<th>Sequence in Phase Window</th>
<th>No. of Optimal Centers Obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measures Based on Maximum Values</td>
<td>(\varphi_q^{\max-n}(30^\circ))</td>
<td>(V-8; A.C.-7; O.C.-11; V.AC.-9)</td>
</tr>
<tr>
<td></td>
<td>(\varphi_q^{\max-n}(10^\circ))</td>
<td>(V-6; A.C.-4; O.C.-12; V.AC.-14)</td>
</tr>
<tr>
<td>Measures Based on Minimum Values</td>
<td>(\varphi_q^{\min-n}(30^\circ))</td>
<td>(V-12; A.C.-12; O.C.-17; V.AC.-15)</td>
</tr>
<tr>
<td></td>
<td>(\varphi_q^{\min-n}(10^\circ))</td>
<td>(V-9; A.C.-7; O.C.-15; V.AC.-13)</td>
</tr>
</tbody>
</table>
Inferences on the Capability of the OLS-RBPNN Modular Version in Classification of Large Dataset PD Signatures

It is evident from this research that the innovative approach of using FOLS-RBPNN for PD pattern classification has the following distinct advantages:

1. FOLS-RBPNN is able to handle a multitude of complexities (varying applied voltage, large and ill-conditioned dataset training and overlapped multiple PD sources) with dexterity.

2. This methodology offers fresh insight into solutions for unique issues related to real-time/ on-line testing and its various associated complexities as compared to any other previously attempted methodologies adopted by researchers including those carried out earlier in this research.

3. Once the training phase of the FOLS-RBPNN is well established, lesser number of exemplars may be used to describe the centers of that class. Further, rapid computation and convergence may be achieved by implementing the ROLS algorithm with a few modifications carried out in the FOLS algorithm since the iterative procedure adopted by the centre selection algorithm is perceptibly fast while it is usually not the case with a few other supervised ANNs.

7.4 Summary and Gist of Major Conclusions

The need for enhanced versions of PNN based on Kernel density estimation for clustering large datasets is discussed. The theoretical foundations related to the formulation of EM algorithm and the ML estimation procedure is deliberated. Concepts
related to the basis for realization of variants of PNN based on single and multiple values of scatter (variance) parameter leading to Homoscedastic and Heteroscedastic PNN respectively are illustrated with suitable algorithms. Another interesting modification involves the aspect of utilizing the Jackknife procedure which enables addressing issues related to large dataset training due to the effect of outliers in the dataset. The formulation of the modified version called the Robust HRPNN (RHRPNN) is also deliberated. Another interesting variant based on subset model selection procedure for center selection (Forward Orthogonal Least Square) has been utilized with two strong features of excellent learning with RBF and exceptional discrimination capability of PNN by implementing the RBPNN variant.

The vital aspects of the performance of the proposed Modular PNN variant are summarized:

1. It is evident from studies that a more parsimonious set of centers is obtained utilizing the HRPNN algorithm with superior classification capability. However, problems related to the effect of outliers in discriminating inter-class clusters is observed to pose some difficulties during classification of patterns pertaining to relatively fully overlapped patterns.

2. Since rapid computation speed and convergence is observed with both the proposed PNN versions (for all the feature vectors) this methodology provides an exciting opportunity for creating a framework for implementing a real-time condition monitoring system for classifying divergent categories of insulation flaws in electrical power apparatus.
3. It is also observed during training of large dataset signatures that the RHRPNN yields an optimal set of PDFs yet being able to counter the influence of outlier exceptionally well. The superiority of the robust procedure in ensuring a much more parsimonious set of centres than the HRPNN in addition to addressing issues pertaining to outliers is made evident during analysis.

4. It is observed that RHRPNN which utilizes the Jackknife estimation procedure which has built-in mechanism for carrying out the hold-one-out validation of the input dataset at a rapid computational speed, reduces computational cost during large dataset training.

5. FOLS-RBPNN is able to handle a multitude of complexities (varying applied voltage, large and ill-conditioned dataset training and overlapped multiple PD sources) with dexterity. Once the training phase of the FOLS- RBPNN is well established, lesser number of exemplars may be used to describe the centres of that class.

6. It is apparent from the investigation and analysis that the Kernel Density Clustering based Modular PNN Ensemble Classifier which include the HOPNN, HRPNN and RHRPNN serves as an excellent framework in training large dataset PD signatures. The results clearly confirm the rapid computational speed, parsimonious set of centers and good class seperability of these variants. The improved classification capability of the HRPNN and RHRPNN is attributed to the inherent strength of the PNN variants in being able to provide separate centers and variances pertaining to sub-classes. Table 7.5 summarizes the classification capability of the Modular PNN version.
Table 7.5: Classification Capability of Enhanced Kernel Clustering based Modular PNN Versions for Large Dataset PD Signature Discrimination

<table>
<thead>
<tr>
<th>Feature Vector</th>
<th>Classification Capability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decision based on Modular HOPNN Version</td>
</tr>
<tr>
<td>Φ-q-n (measures for 10° phase window):</td>
<td>93.4</td>
</tr>
<tr>
<td>1. Measures based on Maximum Values</td>
<td>93.4</td>
</tr>
<tr>
<td>2. Measures based on Statistical Operators &amp; Higher Order Moments</td>
<td>93.4</td>
</tr>
<tr>
<td>3. Measures based on Types of Mean</td>
<td>93.4</td>
</tr>
</tbody>
</table>

7. It is also worth noting that during the analysis of the performance of RHRPNN more frugal number of centers than that of the HRPNN is obtained yet, the classification rate is much improved. This aspect clearly indicates the robustness of the training phase which is inherently imbibed by the RHRPNN structure due to implementation of the Jackknife procedure whose strength lies in its ability to handle the effect of outliers in the signature patterns.
8. The OLS-RBPNN structure also provides a good alternative scheme for the Kernel based Modular PNN variants from the context of providing optimum set of representative centers during the training phase. Though this modular scheme classifies under the sub-space model category and might be grouped into the partitioning based clustering technique, since this methodology is able to cater to large dataset training, the OLS-RPBNN is included as a part of this Modular PNN Version. Improvements in computational capability of this scheme could be obtained by utilizing the Recursive OLS algorithm.