Chapter 11
On-line Condition Monitoring Using Ensemble Learning

11.1 Introduction

In Chap. 3 the Multi-Layer Perceptron (MLP) neural network was introduced for condition monitoring of a population of cylindrical shells. The MLP technique was explained in detail and after a literature review was conducted the technique was implemented to identify faults in a population of cylindrical shells. In that chapter, modal properties and pseudo-modal energies data were applied to classify faults. The principal component analysis method was applied to reduce the dimensions of the input data. The multifold cross-validation method was used to select the optimal number of hidden units amongst the 20 trained pseudo-modal-energy-networks and the 20 trained modal-property-networks. The pseudo-modal-energy-network and the modal-property-network were found to give similar accuracy in classifying faults.

In Chap. 4, two Bayesian multi-layer perceptron neural networks were developed by applying the hybrid Monte Carlo technique, with one trained using pseudo-modal energies while the other was trained using modal properties. They were then applied to the condition monitoring of a population of cylindrical shells. The pseudo-modal-energy-network gave better results than the modal-property-network.

In Chap. 5, a committee of neural networks technique was presented. It applied pseudo modal energies, modal properties and wavelet transform data simultaneously to identify faults in cylindrical shells. The technique was tested to identify faults in a population of ten steel seam-welded cylindrical shells and could identify faults better than the three individual methods.

Next, Chap. 6 extracted bearing vibration signals features using time-domain fractal-based feature extraction. This method applied the Multi-Scale Fractal Dimension (MFD) which was approximated using the Box-Counting Dimension. The extracted features were then used to classify faults using the Gaussian Mixture Models (GMM) and the hidden Markov Models (HMM). The results showed that the feature extraction method revealed fault specific information. Additionally, the experiment demonstrated that HMM outperformed GMM. Nonetheless, the disadvantage of HMM was that it was more computationally expensive to train when compared with the GMM. Consequently, it was concluded that the framework presented gives an improvement in the performance of the bearing fault detection and diagnosis, but it was recommended that the GMM classifier be used when the computational effort is a major issue of consideration.

Chapter 7 presented the application of Fuzzy Set Theory (FST) and fuzzy ARTMAP to diagnose the condition of high voltage bushings. The diagnosis used Dissolved Gas Analysis (DGA) data from bushings based on IEC60599, IEEE C57-104, and the California State University Sacramento (CSUS) criteria for oil impregnated paper (OIP) bushings. FST and fuzzy ARTMAP were compared with regards to accuracy. Both FST and fuzzy ARTMAP could diagnose the bushings condition with 98% and 97.5% accuracy respectively.

Chapter 8 applied the rough set method and the ant colony optimization technique for the condition monitoring of transformer bushings. The theories of rough set and ant colony optimization technique were described and the presented system was tested for the condition monitoring of transformer bushings. The rough set method that was optimized using the ant colony optimization method gave 96.1% accuracy, using 45 rules while the equal-frequency-bin partition model gave 96.4% accuracy, using 206 rules.

In Chap. 9 a technique for fault classification in mechanical systems in the presence of missing data entries was introduced. The technique was based on auto-associative neural networks where the network was trained to recall the input data through some non-linear neural network mapping. From the trained network an error equation with missing inputs as design variables was created. A genetic algorithm was applied to solve for the missing input values. The presented technique was tested on a fault classification problem for a population of cylindrical shells. It was observed that the technique could estimate single-missing-entries to an accuracy of 93% and two-missing-entries to an accuracy of 91%. The approximated values were then applied to the classification of faults and a fault classification accuracy of 94% was observed for single-missing-entry cases and 91% for two-missing-entry cases, while the full database set was able to give a classification accuracy of 96%.

In Chap. 10 feature extraction and condition classification were considered. The feature extraction methods were fractals, Kurtosis and Mel-frequency Cepstral Coefficients. The classification approaches that were applied were the support vector machines (SVM) and extension neural networks (ENN). When applied these techniques gave good results.

Pan et al. (2008) created a remote online machine condition monitoring system which was created using Borland C++ and communication via the internet. A number of signal-processing approaches, for instance time-frequency analysis and order-tracking for signal analysis and pattern recognition were applied using the Borland C++ Builder graphical user interface. The machine fault-diagnostic ability was improved by using the socket application program interface as the transmission control protocol / Internet protocol. The effectiveness of their remote diagnostic system was tested by monitoring a transmission-element test rig and good results were obtained.

Bouhouche et al. (2010) presented a technique for process condition monitoring and evaluation which hybridized the online support vector machine regression and the fuzzy sets approaches. Their technique was based on moving windows so that the past and new data for the model to adapt to the time dependency. A fuzzy analysis was then applied for condition monitoring. Their technique was then applied online to evaluate the quality of a rolling process. The results showed that their technique was simple and gave good results.

Oberholster and Heyns (2009) presented an online condition monitoring technique and applied this to axial-flow turbo-machinery blades. They applied the Eulerian application of laser Doppler vibrometry to accomplish this task. When the method was tested it was found to be viable for the online blade condition monitoring when phase angles at reference frequencies were monitored using a non-harmonic Fourier analysis.

Loutas et al. (2009) presented a condition monitoring system for a single-stage gearbox with induced gear cracks using on-line vibration and acoustic emission measurements. Special attention was paid to the signal processing of the measured vibration and acoustic emission data with the intention of extracting conventional and novel features of diagnostic value from the monitored waveforms. Wavelet-based features used the discrete wavelet transform. The evolution of the chosen features against test time was presented, assessed and the parameters with the most diagnostic characteristics were selected. The advantages of acoustic emission over vibration data for the early diagnosis of natural wear in gear systems were presented.

11.2 Ensemble Methods

The online learning technique implemented in this chapter is based on ensemble learning (Hansen and Salamon 1990; Jordan and Jacobs 1994; Kuncheva et al. 2001). *Ensemble learning* is a technique where multiple models, such as classifiers, are intentionally created and combined to solve a particular problem (Rogova 1994; Polikar 2006). Ensemble learning is usually applied to increase the performance of a model (Xu et al. 1992; Huang and Suen 1993; Dietterich 2000). In this section three ensemble learning approaches are described: bagging, stacking, and adaptive boosting. In particular, the AdaBoost method is described because it was the basis for the creation of the Learn ++ technique, which is the online method adopted for this chapter.

11.2.1 Bagging

Bagging is a technique which is based on the combination of models fitted to randomly selected samples of a training data set to decrease the variance of the prediction model (Efron 1979; Breiman 1996). Bagging basically requires randomly selecting a subset of the training data and using this subset to train a model and repeating this process. Afterwards, all trained models are combined with equal weights to form an ensemble.

11.2.2 Stacking

In the area of modelling, one can choose from a set of models by comparing them using the data that was not used to create the models (Polikar 2006). This prior belief can also be applied to choose a model amongst a set of models, based on a single data set by using a technique called *cross-validation* (Bishop 1995). This is conducted by dividing the data into a *training* data set, which is used to train the models, and a *test* data set. Stacking takes advantage of this prior belief by using the performance from the test data to combine the models instead of choosing among them the best performing model when tested on the test data set (Wolpert 1992).

11.2.3 AdaBoost

Boosting is a method that incrementally creates an ensemble by training each new model with data that the previously trained model misclassified. Then the ensemble, which is a combination of all trained models, is used for prediction.

Adaptive Boosting (AdaBoost) is an extension of boosting to multi-class problems (Freund and Schapire 1997; Schapire et al. 1998). There are many types of AdaBoost, for instance AdaBoost.M1, where each classifier can receive a weighted error of no more than ½, AdaBoost.M2 for those weak classifiers that cannot achieve a weighted error of less than ½.

For AdaBoost.M1, samples are drawn from a distribution D that is updated in such a way that successive classifiers concentrate on difficult cases. This is achieved by adjusting D in such a way that that the earlier, misclassified cases are likely to be present in the following sample. The classifiers are then combined through weighted majority voting. The distribution begins as a uniform distribution so that all cases have equal probability to be drawn into the first data subset S_1.

As described by Polikar (2006), at each iteration t, a new training data subset is sampled, and a weak classifier is trained to create a hypothesis h_t. The error given by this hypothesis with regards to the current distribution is estimated as the sum of distribution weights of the cases misclassified by h_t. AdaBoost.M1 requires that this error is less than ½, and if this requirement is violated then the procedure terminates. The normalized error β_t is then calculated so that the error that is in the [0 0.5] interval is normalized into the [0 1] interval. The transformed error is implemented in the distribution update rule, where $D_t(i)$ is decreased by a factor of $\beta_t, 0 < \beta_t < 1$, if x_i is correctly classified by h_t, or else it is left unaltered. When the distribution is normalized so that $D_{t+1}(i)$ is a proper distribution, the weights of those instances that are misclassified are increased. This update rule guarantees that the weights of all instances correctly classified and the weights of all misclassified instances add up to ½. The requirement for the training error of the base classifier to be less than ½ forces the procedure to correct the error committed by the previous base model. When the training process is complete, the test data are classified by this ensemble of T classifiers, by applying a weighted majority voting procedure where each classifier obtains a voting weight that is inversely proportional to its normalized error (Polikar 2006). The weighted majority voting then selects the class ω allocated the majority vote of all classifiers. The procedure for Adaboost is shown in Algorithm 11.1 (Polikar 2006).

As described by Polikar (2006), the theoretical analysis of the AdaBoost technique shows that the ensemble training error E is bounded above by:

$$E < 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t\,(1 - \varepsilon_t)} \tag{11.1}$$

The $\varepsilon_t < 1/2$ ensemble error E is reduced when new classifiers are added. The AdaBoost is resistant to overfitting, a characteristic that is explained by the margin theory (Schapire 1990; Polikar 2006).

11.3 The Learn++ On-line Method

On-line learning is appropriate for modelling dynamically time-varying systems where the operating conditions change with time. It is also appropriate when the data set available is insufficient and does not completely characterize the system. Another benefit of on-line learning is that it can incorporate new conditions that may be presented by the incoming data.

An on-line bushing condition monitoring system must have incremental learning capability if it is to be used for automatic and continuous on-line monitoring. An on-line bushing monitoring system improves the reliability, diminishes the maintenance cost and minimizes the out-of-service time for a transformer. The basis of on-line learning is incremental learning, which has been studied by a many researchers (Higgins and Goodman 1991; Fu et al. 1996; Yamaguchi et al. 1999; Carpenter et al. 1992). The difficulty in on-line learning is the propensity of an on-line learner to forget the information learned during the initial stages of the learning process (McCloskey and Cohen 1989). The on-line learning technique adopted for this chapter was Learn++ (Polikar et al. 2001).

Vilakazi and Marwala (2007a) applied the on-line incremental learning technique for monitoring the condition of high voltage bushings. Two incremental learning techniques were applied to the problem of condition monitoring. The first technique used was the incremental learning capability of the Fuzzy ARTMAP (FAM), and they investigated whether the ensemble approach can improve the performance

Algorithm 11.1 The AdaBoost algorithm.M1

```
Input:
• Training data $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ with correct labels $\Delta = \{y_1, y_2, ..., y_n\}$
• Weak learn algorithm, known as Weaklearn
• Integer $T$, speciying the number of classifiers

                    $D_1(i) = 1/n; i = 1, ..., n$

For $t = 1,2,\ldots,T$;
  1. Sample a training subset $S_t$, according to the distribution $D_t$
  2. Train Weaklearn with $S_t$, receive hypothesis $h_t : X \to \Delta$
  3. Estimate the error of $h_t : \varepsilon_t = \sum_{i=1}^{n} I\,[h_t(\mathbf{x}_i) \neq y_i] \cdot D_t(i) = \sum_{t:h_t(\mathbf{x}_i)\neq y_i} D_t(i)$ If $\varepsilon_t > \frac{1}{2}$
     terminate.
  4. Estimate the normalized error $\beta_t = \varepsilon_t / (1 - \varepsilon_t) \Rightarrow 0 \leq \beta_t \leq 1$
  5. Update the distribution $D_t$: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t, \text{ if } h_t(\mathbf{x}_i) y_i \\ 1, \text{ otherwise} \end{cases}$ where $Z_t$ is the normalization
     constant so that $D_{t+1}$ becomes a proper distribution function.

Test using majority voting given an unlabeled example $z$ as follows:
  1. Count the total vote from the classifiers $V_j = \sum_{t:h_t(z)} \log(1/\beta_t) j = 1, ..., C$
  2. Select the class that receives the highest number of votes as the final classification.
```

of the FAM. The second technique applied was Learn++ that implemented an ensemble of the multi-layer perceptron classifiers. Both methods were performed well when tested for transformer bushing condition monitoring.

Mohamed et al. (2007) applied incremental learning for the classification of protein sequences. They used the fuzzy ARTMAP as an alternative machine learning system with the ability to incrementally learn new data as it becomes available. The fuzzy ARTMAP was seen to be comparable to many other machine learning systems. The application of an evolutionary strategy in the selection and combination of individual classifiers into an ensemble system, coupled with the incremental learning capability of the fuzzy ARTMAP was shown to be suitable as a pattern classifier. Their algorithm was tested using the data from the G-Coupled Protein Receptors Database and it demonstrated a good accuracy of 83%.

Mohamed et al. (2006) applied fuzzy ARTMAP for multi-class protein sequence classification. They presented a classification system that used pattern recognition method to produce a numerical vector representation of a protein sequence and then classified the sequence into a number of given families. They applied fuzzy ARTMAP classifiers and showed that, when coupled with a genetic algorithm based feature subset selection, the system could classify protein sequences with an accuracy of 93%. This accuracy was then compared to other classification techniques and it was shown that the fuzzy ARTMAP was most suitable because of its high accuracy, quick training times and ability to incrementally learn.

Perez et al. (2010) applied a population-based, incremental learning approach to microarray gene expression feature selection. They evaluated the usefulness of the Population-Based Incremental Learning (PBIL) procedure in identifying a class differentiating gene set for sample classification. PBIL was based on iteratively evolving the genome of a search population by updating a probability vector, guided by the extent of class-separability demonstrated by a combination of features. The PBIL was then compared to standard Genetic Algorithm (GA) and an Analysis of Variance (ANOVA) method. The procedures were tested on a publically available three-class leukaemia microarray data set ($n = 72$). After running 30 repeats of both GA and PBIL, the PBIL could identify an average feature-space separability of 97.04%, while GA achieved an average class-separability of 96.39%. The PBIL also found smaller feature-spaces than GA, (PBIL – 326 genes and GA – 2652) thus excluding a large percentage of redundant features. It also, on average, outperformed the ANOVA approach for $n = 2652$ (91.62%), $q < 0.05$ (94.44%), $q < 0.01$ (93.06%) and $q < 0.005$ (95.83%). The best PBIL run (98.61%) even outperformed ANOVA for $n = 326$ and $q < 0.001$ (both 97.22%). PBIL's performance was credited to its ability to direct the search, not only towards the optimal solution, but also away from the worst.

Hulley and Marwala (2007) applied GA-based incremental learning for optimal weight and classifier selection. They then compared Learn++, which is an incremental learning algorithm to the new Incremental Learning Using Genetic Algorithm (ILUGA). Learn++ demonstrated good incremental learning capabilities on benchmark datasets on which the new ILUGA technique was tested. ILUGA showed good incremental learning ability using only a few classifiers and did not suffer from catastrophic forgetting. The results obtained for ILUGA on the Optical Character Recognition (OCR) and Wine datasets were good, with an overall accuracy of 93% and 94% respectively showing a 4% improvement over Learn++.MT for the difficult multi-class OCR dataset.

Lunga and Marwala (2006a) applied a time series analysis using fractal theory and on-line ensemble classifiers to model the stock market. The fractal analysis was implemented as a concept to identify the degree of persistence and self-similarity within the stock market data. This concept was carried out using the Rescaled range analysis (R/S) technique. The R/S analysis outcome was then applied to an on-line incremental algorithm (Learn++) that was built to classify the direction of movement of the stock market. The use of fractal geometry in this study provided a way of determining quantitatively the extent to which the time series data could be predicted. In an extensive test, it was demonstrated that the R/S analysis provided a very sensitive technique to reveal hidden long runs and short run memory trends within the sample data. A time series data that was measured to be persistent was used in training the neural network. The results from the Learn++ algorithm showed a very high level of confidence for the neural network to classify sample data accurately.

Lunga and Marwala (2006b) applied incremental learning for the on-line forecasting of stock market movement direction. In particular, they presented a specific application of the Learn++ algorithm, and investigated the predictability of financial movement direction with Learn++ by forecasting the daily movement direction of the Dow Jones. The framework was implemented using the Multi-Layer Perceptron (MLP) as a weak learner. First, a weak learning algorithm, which tried to learn a class concept with a single input perceptron, was established. The Learn++ algorithm was then applied to improve the weak MLP learning capacity and thus introduced the concept of incremental on-line learning. The presented framework could adapt as new data were introduced and could classify the data well.

Vilakazi and Marwala (2007b) applied incremental learning to bushing condition monitoring. They presented a technique for bushing fault condition monitoring using the fuzzy ARTMAP. The fuzzy ARTMAP was introduced for bushing condition monitoring because it can incrementally learn information as it becomes available. An ensemble of classifiers was used to improve the classification accuracy of the systems. The test results showed that the fuzzy ARTMAP ensemble gave an accuracy of 98.5%. In addition, the results showed that the fuzzy ARTMAP could update its knowledge in an incremental fashion without forgetting the previously learned information.

Nelwamondo and Marwala (2007) successfully applied a technique for handling missing data from heteroskedastic and non-stationary data. They presented a computational intelligence approach for predicting missing data in the presence of concept drift using an ensemble of multi-layered feed-forward neural networks. Six instances prior to the occurrence of missing data were used to approximate the missing values. The algorithm was applied to a simulated time series data sets that resembled non-stationary data from a sensor. Results showed that the prediction of missing data in a non-stationary time series data was possible but was still a challenge. For one test, up to 78% of the data could be predicted within a 10% tolerance range of accuracy.

Other successful implementations of incremental learning techniques include its use in anomaly detection (Khreich et al. 2009), in human robot interaction (Okada et al. 2009), for online handwriting recognition (Almaksour and Anquetil 2009), for predicting human and vehicle motion (Vasquez et al. 2009) and in visual learning (Huang et al. 2009).

11.3.1 Learn++

Learn++ is an incremental learning algorithm that was introduced by Polikar and co-workers (Polikar et al. 2000, 2001, 2002; Muhlbaier et al. 2004; Erdem et al. 2005; Polikar 2006). It is based on AdaBoost and applies multiple classifiers to enable the system to learn incrementally. The algorithm operates on the concept of using many classifiers that are weak learners to give a good overall classification. The weak learners are trained on a separate subset of the training data and then the classifiers are combined using a weighted majority vote. The weights for the weighted majority vote are chosen using the performance of the classifiers on the entire training dataset.

Each classifier is trained using a training subset that is drawn according to a specified distribution. The classifiers are trained using a weak learn algorithm (WeakLearn). The requirement for the WeakLearn algorithm is that it must give a classification rate of less than 50% initially (Polikar et al. 2002). For each database *Dk* that contains training sequence, *S*, where *S* contains learning examples and their corresponding classes, Learn++ starts by initializing the weights, *w*, according to a specified distribution *DT,* where *T* is the number of hypothesis. Firstly the weights are initialized to be uniform, thereby giving equal probability for all cases selected for the first training subset and the distribution is given by (Polikar et al. 2002):

$$D = {}^{1}\!/_{m} \tag{11.2}$$

Here, *m* represents the number of training examples in *S*. The training data are then divided into training subset *TR* and testing subset *TE* to ensure the WeakLearn capability. The distribution is then used to select the training subset *TR* and testing subset *TE* from *Sk*. After the training and testing subset have been selected, the WeakLearn algorithm is implemented. The WeakLearner is trained using subset *TR*. A hypothesis, h_t, obtained from a WeakLearner is tested using both the training and testing subsets to obtain an error (Polikar et al. 2002):

$$\varepsilon_t = \sum_{t:h_i(x_i)\neq y_i} D_t(i) \tag{11.3}$$

The error is required to be less than 0.5; a normalized error βt is computed using (Polikar et al. 2002):

$$B_t = {}^{\varepsilon_t}\!/_{1-\varepsilon_t} \tag{11.4}$$

If the error is greater than 0.5, the hypothesis is discarded and a new training and testing subsets are selected according to a distribution D_T and another hypothesis is computed. All classifiers generated are then combined using weighted majority voting to obtain the composite hypothesis, H_t (Polikar et al. 2002):

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log\left({}^{1}\!/_{\beta_t}\right) \tag{11.5}$$

Weighted majority voting gives higher voting weights to a hypothesis that performs well on the training and testing data subsets. The error of the composite hypothesis is computed as follows (Polikar et al. 2002):

$$E_t = \sum_{t:H_i(x_i)\neq y_i} D_t(i) \tag{11.6}$$

If the error is greater than 0.5, the current hypothesis is discarded and the new training and testing data are selected according to the distribution D_T. Otherwise, if the error is less than 0.5, then the normalized error of the composite hypothesis is computed as follows (Polikar et al. 2002):

$$B_t = {}^{E_t}\!/1 - E_t \tag{11.7}$$

The error is used in the distribution update rule, where the weights of the correctly classified case are reduced, consequently increasing the weights of the misclassified instances. This ensures that the cases that were misclassified by the current hypothesis have a higher probability of being selected for the subsequent training set. The distribution update rule is given by the following equation (Polikar et al. 2002):

$$w_{t+1} = w_t(i) \times B_t^{1-[|H_t(x_i)\neq y_i|]} \tag{11.8}$$

After the T hypothesis has been created for each database, the final hypothesis is computed by combining the hypotheses using weighted majority voting as described by the following equation (Polikar et al. 2002):

$$H_t = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:H_t(x)=y} \log\left({}^{1}\!/_{\beta_t}\right) \tag{11.9}$$

The Learn++ algorithm is represented diagrammatically in Fig. 11.1.

11.3.2 Confidence Measurement

A technique is used to estimate the confidence of the algorithm about its own decision. A majority of hypotheses agreeing on given instances can be interpreted as an indication of confidence on the decision proposed. If it is assumed that a total of T hypotheses are generated in k training sessions for a C-class problem, then for any given example, the final classification class, the total vote class c receives is given by (Muhlbaier et al. 2004):

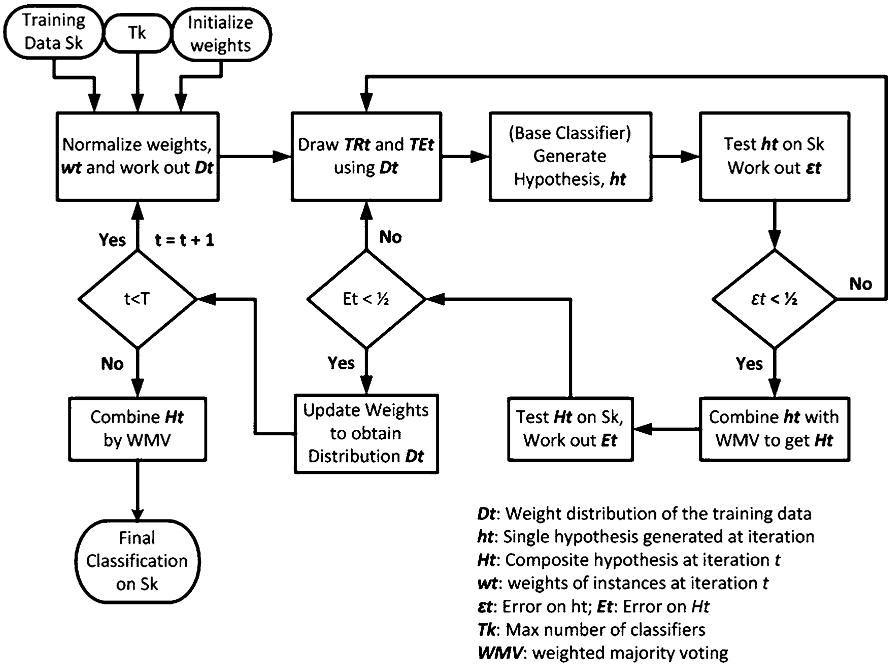
Training Data Sk
Tk
Initialize weights
Normalize weights, wt and work out Dt
Draw TRt and TEt using Dt
(Base Classifier) Generate Hypothesis, ht
Test ht on Sk Work out εt
Yes
t = t + 1
No
t<T
Et < ½
εt < ½
No
Yes
No
Yes
Combine Ht by WMV
Update Weights to obtain Distribution Dt
Test Ht on Sk, Work out Et
Combine ht with WMV to get Ht
Final Classification on Sk
Dt: Weight distribution of the training data
ht: Single hypothesis generated at iteration
Ht: Composite hypothesis at iteration t
wt: weights of instances at iteration t
εt: Error on ht; Et: Error on Ht
Tk: Max number of classifiers
WMV: weighted majority voting

Fig. 11.1 Block diagram of a Learn++ algorithm

$$\zeta_c = \sum_{t:h_t(\mathbf{x})=c} \Psi_t \tag{11.10}$$

where Ψt denotes the voting weights of the t^{th}, hypothesis h_t.

Normalizing the votes received by each class can be performed as follows (Muhlbaier et al. 2004):

$$\lambda_c = \frac{\zeta_c}{\sum\limits_{c=1}^{C} \zeta_c} \tag{11.11}$$

Here, λc can be interpreted as a measure of confidence on a scale of 0–1. A high value of λc shows high confidence in the decision and conversely, a low value of λc shows low confidence in the decision. It should be noted that the λc value does not represent the accuracy of the results, but the confidence of the system in its own decision.

11.4 Multi-Layer Perceptrons

The architecture considered in this chapter to create the WeakLearn was the multi-layer perceptron (MLP) as described in great detail in Chap. 3. The MLP can be defined as a feed-forward neural network model that approximates the relationship between a set of input data and a set of appropriate output data. Its foundation is the standard linear perceptron. It makes use of three or more layers of neurons usually with non-linear activation functions. This is because it can distinguish data that are not linearly separable, or separable by a hyper-plane. The multi-layer perceptron has been used to model many complex systems in areas such as mechanical and aerospace engineering as well as for modelling interstate conflict (Marwala 2007; Marwala 2009; Marwala 2010; Marwala and Lagazio 2011).

The MLP neural network consists of multiple layers of computational units, usually inter-connected in a feed-forward way (Bishop 1995). Each neuron in one layer is directly connected to the neurons of the subsequent layer. A fully connected two-layered MLP architecture was used for this chapter. A two-layered MLP architecture was used because of the universal approximation theorem, which states that a two-layered architecture is adequate for MLP and, consequently, it can approximate data of arbitrary complexity (Bishop 1995).

11.5 Experimental Investigation

A dissolved gas analysis is used to estimate the faulty gases in bushing oil. The information from the dissolved gas analysis reflects the states of the transformer and bushing. Ten diagnostic gases are extracted: CH_4, C_2H_6, C_2H_4, C_2H_2, H_2, CO, CO_2, N_2, O_2 and total dissolved combustible gases. The total dissolved combustible gas is given by the sum of methane, hydrogen, acetylene, ethane, ethylene and hydrogen. The faulty gases are analyzed using the IEEE C57.104 standards. Data pre-processing is an integral part of neural network architecture. Data pre-processing makes it easier for the network to learn. Data are normalized to fall within 0 and 1.

The first experiment evaluated the incremental capability of the Learn++ algorithm using a first-level fault diagnosis, which was aimed at classifying the presence or the absence of faults in transformer bushings. The data used were collected from bushings over a period of 2.5 years from bushings in service. The algorithm was implemented with 1,500 training examples and 4,000 validation examples. The training data were divided into five databases each with 300 training instances. In each training session, Learn++ was provided with each database and 20 hypotheses were generated. The WeakLearner used an MLP with 10 input layer neurons, 5 hidden layer neurons and one output layer neuron. To ensure that the technique retained previously learned data, the previous database was tested at each training session.

Table 11.1 Performance of Learn++ for first level online condition monitoring (Key: $S =$ dataset)

Dataset	S_1	S_2	S_3	S_4	S_5
S_1	89.5	85.8	83.0	86.9	85.3
S_2	–	91.4	94.2	93.7	92.9
S_3	–	–	93.2	90.1	91.4
S_4	–	–	–	92.2	94.5
S_5	–	–	–	–	98.0
Learn ++ Testing (%)	65.7	79.0	85.0	93.5	95.8

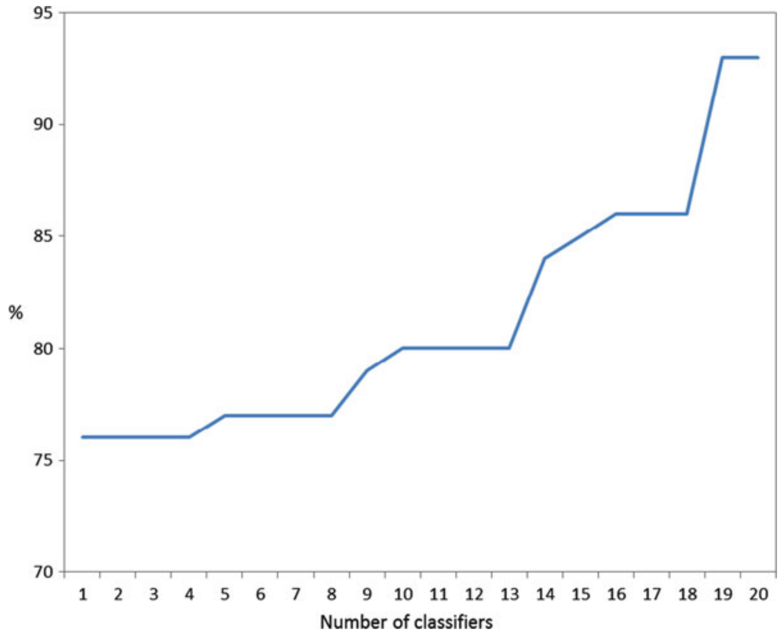

Fig. 11.2 Performance of Learn++ on training data against the number of classifiers

The first row of Table 11.1 shows the performance of Learn++ on the training data for different databases. On average, the WeakLearner gave 60% classification rate on its training dataset, which improved to 98% when the hypotheses were combined.

These results show the performance improvement of Learn++ with a single database. Each column shows the performance of current and previous databases. This is to indicate that Learn++ did not forget the previously learned information when new data were introduced.

The classifiers' performance on the testing dataset steadily increased from 65.7% to 95.8% as new databases became available, demonstrating the incremental capability of Learn++ as shown in Fig. 11.2.

A second experiment was performed to evaluate whether the frameworks can accommodate new classes. The results appear in Table 11.2. The faulty data

Table 11.2 Performance of Learn++ for second stage bushing condition monitoring

Dataset	S_1	S_2	S_3	S_4	S_5
S_1	95.0	95.2	94.6	95.7	95.1
S_2	–	96.3	96.0	96.8	95.3
S_3	–	–	97.0	96.4	96.5
S_4	–	–	–	97.8	96.8
S_5	–	–	–	–	99.2
Learn ++ Testing (%)	60.0	65.2	76.0	83.0	95.3

were divided into 1,000 training examples and 2,000 validation examples, which contained all three classes. The training data were divided into five databases, each with 200 training instances. The first and second databases contained training examples of partial discharges and thermal faults.

The data with unknown faults were introduced in training session three. In each training session, Learn++ was provided with each database and 20 hypotheses were generated. The classifiers performance increased from 60% to 95.3% as new classes were introduced in subsequent training datasets. The final experiment addressed the problem of bushing condition monitoring using a MLP network that was trained using batch learning. This was done to compare the classification rate of Learn++ with that of a MLP.

A MLP with the same set of training example as Learn++ was trained and the trained MLP was tested with the same validation data as Learn++. This test was conducted for the first and second levels of fault classification. In the first level fault diagnosis, the MLP gave a classification rate of 97.2% whereas the second level MLP gave a classification rate of 96.0%. This was when the classifier had seen all the fault classes *a priori*. If the classifier had not seen all the fault cases, the performance decreased from 65.7% for database 1–30.0% for databases 2–3 for the first level fault classification.

11.6 Conclusion

This chapter presented an on-line bushing condition monitoring approach, which can adapt to newly acquired data. This technique was capable of factoring into account new classes that were introduced by incoming data and was implemented using an incremental learning algorithm that used the MLP called Learn++. The test results improved from 67.5% to 95.8% as new data were introduced and improved from 60% to 95.3% as new conditions were introduced. On average, the confidence value of the framework about its decision was 0.92.

References

Almaksour A, Anquetil E (2009) Fast incremental learning strategy driven by confusion reject for online handwriting recognition. In: Proceedings of the international conference on document analysis and recognition, Barcelona, Spain, pp 81–85

Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford

Bouhouche S, Yazid LL, Hocine S, Bast J (2010) Evaluation using online support-vector-machines and fuzzy reasoning. Application to condition monitoring of speeds rolling process. Control Eng Pract 18:1060–1068

Breiman L (1996) Bagging predictors. Mach Learn 24:123–140

Carpenter GA, Grossberg S, Marhuzon N, Reynolds JH, Rosen DB (1992) ARTMAP: a neural network architecture for incremental learning supervised learning of analog multidimensional maps. IEEE Trans Neural Netw 3:698–713

Dietterich TG (2000) Ensemble methods in machine learning. Lect Notes Comput Sci 1857:1–15

Efron B (1979) Bootstrap methods: another look at the jackknife. Ann Stat 7:1–26

Erdem Z, Polikar R, Gurgen F, Yumusak N (2005) Reducing the effect of out-voting problem in ensemble based incremental support vector machines. Lect Notes Comput Sci 3697:607–612

Freund Y, Schapire RE (1997) Decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55:119–139

Fu L, Hsu HH, Principe JC (1996) Incremental backpropagation networks. IEEE Trans Neural Netw 7:757–761

Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 12:993–1001

Higgins CH, Goodman RM (1991) Incremental learning for rule based neural network. In: Proceedings of the internatinal joint conference on neural networks, Seattle, Washington, pp 875–880

Huang YS, Suen CY (1993) Behavior-knowledge space method for combination of multiple classifiers. In: Proceedings of the IEEE computer vision and pattern recognition conference, New York, USA, pp 347–352

Huang D, Yi Z, Pu X (2009) A new incremental PCA algorithm with application to visual learning and recognition. Neural Process Lett 30:171–185

Hulley G, Marwala T (2007) Genetic algorithm based incremental learning for optimal weight and classifier selection. In: Proceedings of the AIP conference, Vancouver, Canada, pp 258–267

Jordan MJ, Jacobs RA (1994) Hierarchical mixtures of experts and the EM algorithm. Neural Comput 6:181–214

Khreich W, Granger E, Miri A, Sabourin RA (2009) A comparison of techniques for on-line incremental learning of HMM parameters in anomaly detection. In: Proceedings of the IEEE symposium on computational intelligence for security and defense applications, Ottawa, pp 1–8

Kuncheva LI, Bezdek JC, Duin R (2001) Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recognit 34:299–314

Loutas TH, Sotiriades G, Kalaitzoglou I, Kostopoulos V (2009) Condition monitoring of a single-stage gearbox with artificially induced gear cracks utilizing on-line vibration and acoustic emission measurements. Appl Acoust 70:1148–1159

Lunga D, Marwala T (2006a) Online forecasting of stock market movement direction using the improved incremental algorithm. Lect Notes Comput Sci 4234:440–449

Lunga D, Marwala T (2006b) Time series analysis using fractal theory and online ensemble classifiers. Lect Notes Comput Sci 4304:312–321

Marwala T (2007) Computational intelligence for modelling complex systems. Research India Publications, Delhi

Marwala T (2009) Computational intelligence for missing data imputation, estimation and management: knowledge optimization techniques. IGI Global Publications, New York

Marwala T (2010) Finite element model updating using computational intelligence techniques. Springer, London

Marwala T, Lagazio M (2011) Militarized conflict modeling using computational intelligence techniques. Springer, London

McCloskey M, Cohen N (1989) Catastrophic interference connectionist networks: the sequential learning problem. Psychol Learn Motiv 24:109–164

Mohamed S, Rubin D, Marwala T (2006) Multi-class protein sequence classification using fuzzy ARTMAP. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, Taipei City, pp 1676–1681

Mohamed S, Rubin D, Marwala T (2007) Incremental learning for classification of protein sequences. In: Proceedings of the IEEE international joint conference on neural networks, Orlando, Florida, pp 19–24

Muhlbaier M, Topalis A, Polikar R (2004) Learn++.MT: a new approach to incremental learning. Lect Notes Comput Sci 3077:52–61

Nelwamondo FV, Marwala T (2007) Handling missing data from heteroskedastic and nonstationary data. Lect Notes Comput Sci 4491:1293–1302

Oberholster AJ, Heyns PS (2009) Online condition monitoring of axial-flow turbomachinery blades using rotor-axial Eulerian laser Doppler vibrometry. Mech Syst Signal Process 23: 1634–1643

Okada S, Kobayashi Y, Ishibashi S, Nishida T (2009) Incremental learning of gestures for human-robot interaction. AI Soc 25:155–168

Pan MC, Li PC, Cheng YR (2008) Remote online machine condition monitoring system. Measurement 41:912–921

Perez M, Featherston J, Marwala T, Scott LE, Stevens DM (2010) A population-based incremental learning approach to microarray gene expression feature selection. In: Proceedings of the IEEE 26th convention of electrical and electronic engineers, Eilat, Israel, pp 10–14

Polikar R (2006) Ensemble based systems in decision making. IEEE Circuit Syst Mag 6:21–45

Polikar R, Udpa L, Udpa S, Honavar V (2000) Learn++: an incremental learning algorithm for multilayer perceptrons. In: Proceedings of IEEE 25th international conference on acoustics, speech and signal processing, Michigan, USA, pp 3414–3417

Polikar R, Udpa L, Udpaand S, Honavar V (2001) Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cybern 31:497–508

Polikar R, Byorick J, Krause S, Marino A, Moreton M (2002) Learn++: a classifier independent incremental learning algorithm for supervised neural networks. In: Proceedings of the international joint conference on neural networks, Honolulu, pp 1742–1747

Rogova G (1994) Combining the results of several neural network classifiers. Neural Netw 7: 777–781

Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227

Schapire RE, Freund Y, Bartlett P, Lee WS (1998) Boosting the margin: a new explanation for the effectiveness of voting methods. Ann Stat 26:51–1686

Vasquez D, Fraichard T, Laugier C (2009) Growing hidden Markov models: an incremental tool for learning and predicting human and vehicle motion. Int J Robot Res 28:1486–1506

Vilakazi CB, Marwala T (2007a) Incremental learning and its application to bushing condition monitoring. Lect Notes Comput Sci 4491:1237–1246

Vilakazi CB, Marwala T (2007b) Online incremental learning for high voltage bushing condition monitoring. In: Proceedings of the international joint conference on neural networks, Orlando, Florida, pp 2521–2526

Wolpert DH (1992) Stacked generalization. Neural Netw 5:241–259

Xu L, Krzyzak A, Suen CY (1992) Methods for combining multiple classifiers and their applications to handwriting recognition. IEEE Trans Syst Man Cybern 22:418–435

Yamaguchi K, Yamaguchi N, Ishii N (1999) Incremental learning method with retrieving of interfered patterns. IEEE Trans Neural Netw 10:1351–1365