Pattern Recognition Coursework Two

Lizhang Lin (CID: 00840705) & Lida Jin (CID: 00835494)
Department of Electrical and Electronic Engineering
Imperial College London

1. Distance Metrics

1.1. Analyse Data Distributions

We were given wine recognition data with three classes of 178 features and each feature contains 13 dimensions. The data were randomly split into three sets for training (118), validation (20) and testing (40), in which validation data was applied to find an appropriate model. The test data was used to test and find the accuracy of the selected model.

1.2. Raw Data Distribution

To understand the distributions of training and validation data, means and covariance matrices of three classes were computed.

Figure 1 shows the covariance between various dimensions for different classes. The covariance matrix is relatively flat except that between dimension 13 and any other dimensions. To better understand the reason behind this, we then plotted Figure 3 (log scale) which illustrates the variance and mean of different dimensions of the raw data. We found that the variance of dimension 13 is extremely large for all three classes. This is because that raw data is scaled at dimension 13 as shown in Figure 2.

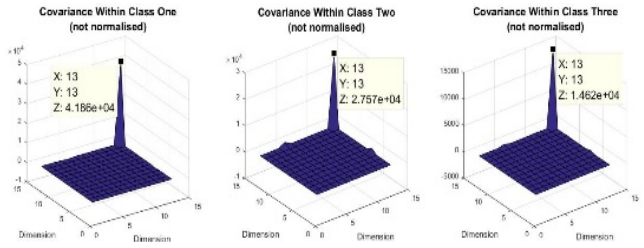

Figure 1 Covariance matrices within all three classes

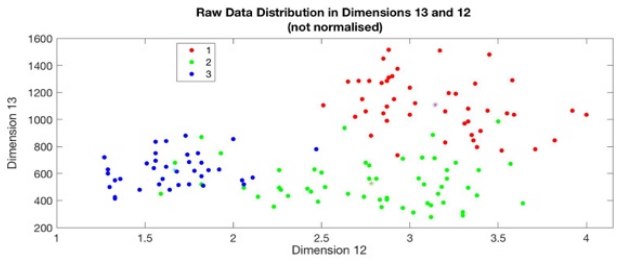

Figure 2 Raw Data Distribution in Dimension 13 and 12

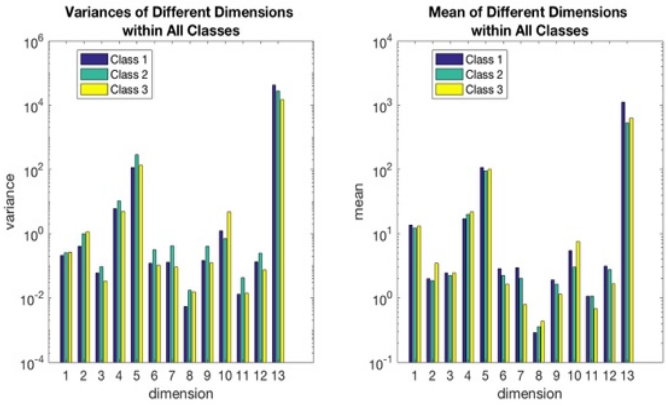

Figure 3 Variance and Mean of Different Dimensions of Raw Data

1.3. Normalised Data Distribution

To make sure that the coefficients of a model are not scaled with respect to the units of the inputs, all the dimensions of the raw data were normalised to unit norm L2.

The covariance matrices of various dimensions within different classes are illustrated in Figure 4. Figure 5 shows the variances and means of normalised data at different dimensions. It is obvious that variances of normalised data are of orders of magnitude smaller than that of raw data. Also, it should be noted that variance of data at dimension 13 no longer dominates the features. Thus, normalisation removes scaling of units and allow other dimensions to play a bigger role.

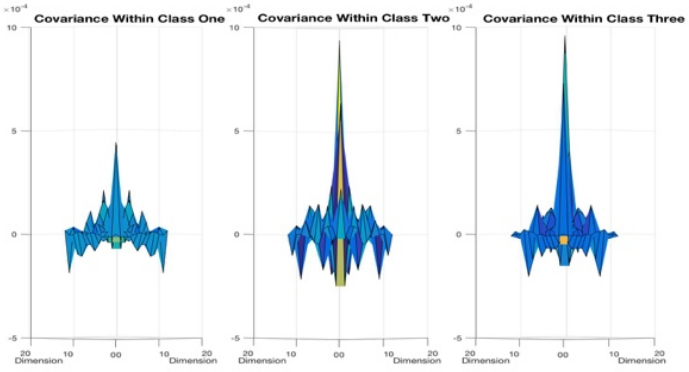

Figure 4 Covariance of Normalised Data Within Different Classes

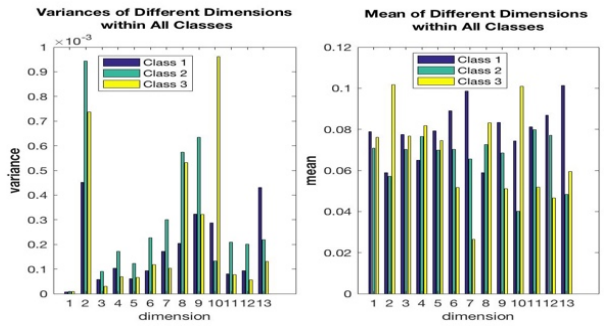

Figure 5 Variance and Mean of Normalised Data at Different Dimensions

1.4. Nearest Neighbour (NN) Classification

Test data was identified by assigning the label of the training data which were found using Nearest Neighbour (NN) classification method. Different distance metrics were used, including L2, L1, Chi2, Histogram Intersection, Correlation and Mahalanobis.

Figure 6 shows error rates of NN classification for both raw data and normalised data with different metrics. The computed error rates are averaged results over a hundred randomly selected test and training data set. Four types of Mahalanobis distance metrics, 'Maha1', 'Maha2', 'Maha3' and 'Maha', were computed with different covariance matrices, namely covariance matrices of features from class 1, 2, 3, and all data respectively. Detailed analyses can be found in the following subsections.

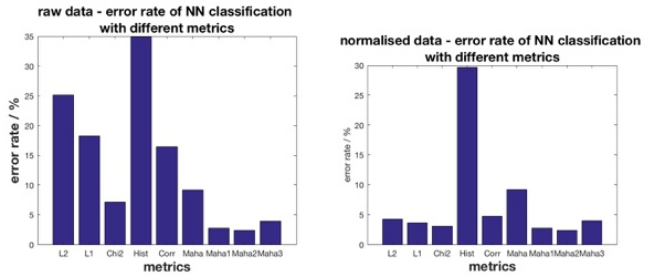

Figure 6 Classification Error of NN Classification with Different Metrics using Raw Data (left) and Normalised Data (right)

1.4.1. L2 & L1 Distance Metrics

L2 and L1 distances are computed by equations

$$L2(Q,T) = \sqrt{\sum_i (Q_i - T_i)^2} \quad (1)$$

$$L1(Q,T) = \sum_i |(Q_i - T_i)| \quad (2)$$

Where Q is the query data, T is the test data, and i stands for i^{th} dimension.

Figure 7 shows failure classifications using Nearest Neighbour method with metrics L1 and L2. As per our findings on raw data distribution in section 1.2, raw data at dimension 13 is much more dispersed than at other dimensions. Thus, most of failure predictions lie in the region where raw data of all classes at dimension 13 are mixed (see the 'X's in figure 7).

It should be noted that L1 distance metric gives less error rate for both normalised and unnormalised features. Equation (2) states that L1 has reduced exponent compared to L2. This makes other features play a bigger role by lowering the significance of a high difference in some given dimension, e.g. dimension 13 of raw data. Thus, L1 is a better distance metric than L2 for NN classification of data at high dimensions.

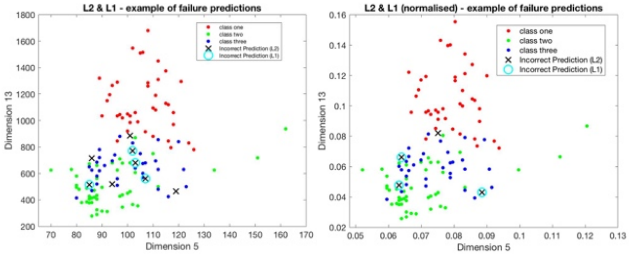

Figure 7 Example of Failure Predictions with Un-normalised (left) and Normalised Data (right)

1.4.2. Chi2 Distance Metric

The formula of Chi-square distance:

$$Chi2(Q,T) = \frac{1}{2}\sum_i \frac{(Q_i - T_i)^2}{(Q_i + T_i)} \quad (3)$$

As shown in the expression, squared L2 distance at each dimension is weighted by the sum of the distances of test and training data from origin. Therefore, when dealing with raw data, less error rate can be achieved as compared to that with L2.

1.4.3. Histogram Intersection

Histogram Intersection is computed by equation

$$Hist(Q,T) = \frac{\sum_i \min(Q_i, T_i)}{\sum_i Q_i} \quad (4)$$

$$d = 1 - Hist(Q,T) \quad (5)$$

The result of equation (4) is within range [0,1] where 1 stands for highest similarity. In our experiment, Histogram Intersection metric yields worst error rate. It cannot tell the similarity between test and training data when the value of the training data is greater than that of the test data.

1.4.4. Correlation Distance Metric

$$corr(Q,T) = \frac{cov(Q,T)}{\sigma_Q \sigma_T} \quad (6)$$

$$d = 1 - corr(Q,T) \quad (7)$$

The Correlation distance metric quantifies how well the patterns of two data objects fit into each other. A significant benefit is that it corrects for scaling of different dimensions.

1.4.5. Mahalanobis Distance Metric

Mahalanobis distance can be viewed as the squared Euclidean distance after applying the linear transformation G (see equation 8).

$$d_{Maha}(Q,T) = [G(Q-T)]'[G(Q-T)] \quad (8)$$

It measures how many standard deviations away a training data is from the test data. Our results show that Mahalanobis distance metric has the best performance (least error rate) over all other metrics. Figure 8 is an example that compares Mahalanobis distance with Euclidean distance. The eclipse contour is the Mahalanobis distance which is weighted by covariance matrix of class one. In contrast, Euclidean distance is not a good metric for the example as the data distribution is spread along the positive diagonal.

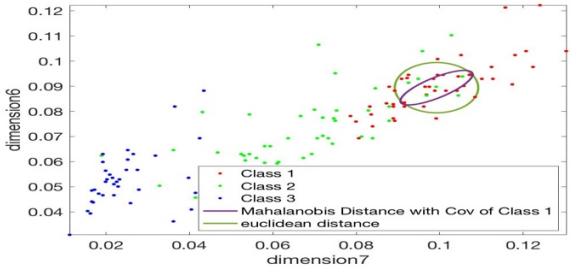

Figure 8 Distribution of Normalised Data at Dimension 6 & 7 with Examples with Mahalanobis Distance and Euclidean Distance

2. NN Classification after K-means Clustering

K-means clustering was applied on normalised training features with different metrics, including Square Euclidean, Cityblock, Cosine, Correlation and Mahalanobis with different covariance matrices. K different clustering centres were generated with distinct class labels. After clustering, classification was performed using NN method with the same distance metric used for clustering.

2.1. Avoid Local Minimum

Note that we used MATLAB K-means algorithm for clustering. The solution that the algorithm reaches often depends on the starting points. It is possible that it sometimes returns a local optimum result. Therefore, we repeated each K-means algorithm a hundred times and take the minimum to obtain a result approximate to global optimum. In addition, we repeated the experiment sufficient number of times with different training and test data sets to compute averaged error rates of different metrics.

2.2. Mapping Clusters to Classes

K-means algorithm generates K number of new clusters. NN classification finds a nearest cluster for each test data. Each cluster maps to a class. If a predicted class points to the class label of the test data, then it is a correct prediction. Otherwise, the prediction is incorrect.

Essentially, in each cluster, there are at most three classes of training data. The class that has the largest population in the cluster is mapped by the cluster.

2.3. Error Rates

The technique we used to map clusters to classes inherently introduces disagreements between the original classes in training data and the classes after being mapped by clusters generated from K-means. We name this type of difference as error of K-means clustering.

NN classification was performed to find the nearest cluster that a test data belongs to. This introduces the second source of error which is relatively small as per our findings in section 1.

2.4. K=3

We tried 5 different types of metrics for both K-means clustering and NN classification to find the nearest cluster. Note that ‘Maha’, ‘Maha1’, ‘Maha2’ and ‘Maha3’ stands for Mahalanobis distance metrics with covariance matrices of all training data, class one training data, class two training data and class three training data respectively. Figure 9 shows the averaged error rates of classification. Figure 10 shows the averaged error rates of K-means clustering.

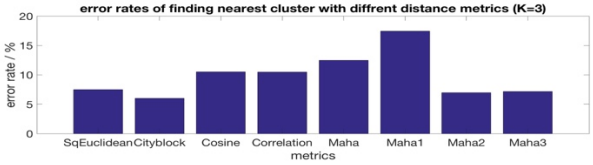

Figure 9 Averaged Error Rates of Classification after K-means

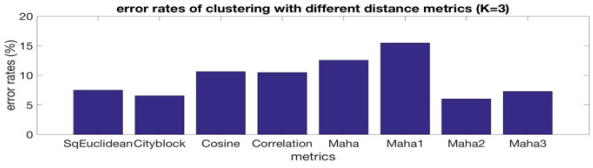

Figure 10 Averaged Error Rates of K-means Clustering

2.4.1. Mahalanobis

It is obvious from Figure 9 that ‘Maha2’ and ‘Maha3’ have good performance. However, ‘Maha1’ gives the largest averaged error rates. This is because the data clustering after K-means disagrees much with the original training classes (see ‘Maha1’ in Figure 10).

We provided an example of data distributions to explain the large error rate of 'Maha1' . Figure 11 (left) shows the data distribution after Mahalanobis transformation G (see Equation 8) on normalised training data. Figure 11 (right) illustrates the distribution of the transformed training data after K-means clustering. It is obvious to find misclustering of data after K-means. E.g. many data in Class 1 (red) are clustered into a new cluster which maps to Class 3 (blue). As a result, the computed error rate of the example with 'Maha1' distance matric is as large as 35%.

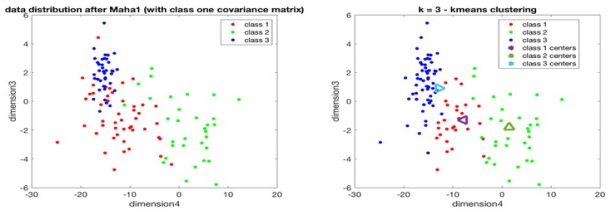

Figure 11 Example – Classification Error Rate 35%.

2.4.2. Cityblock & Square Euclidean Metrics

Cityblock distance metric is again better than Euclidean for data classification/clustering in high dimension because it lowers the significances of difference in some given dimension such that other features play a bigger role.

2.4.3. Cosine & Correlation Distance Metrics

Cosine and Correlation distance metrics can both be viewed as variants in the inner product. Cosine similarity measures the inner products between two vectors divided by their L2 norms. Correlation is the cosine of centred versions of two vectors. Thus, they return a similar error rates. However, there is a drawback of both metrics – they only measure the similarity in terms of angles but ignore the distance between points. Thus, they have similar accuracy as compared to L1 and L2 which only measure the similarity in terms of distance but ignore the angle between vectors.

2.5. K=10

Note that when K=10, at least one cluster is mapped to a class in training data. Figure 12 shows the averaged error rates of finding the nearest cluster with different distance metrics.

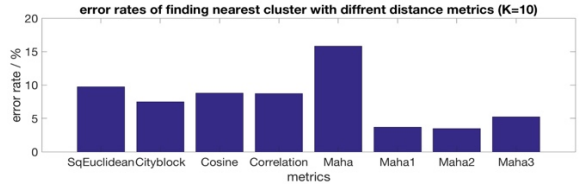

Figure 12 Averaged Error Rates of Classification after K-means

Compared to the results of K=3, there is a significant drop in the error rate of 'Maha1'. However, the averaged error rate of 'Maha' increased to 16%.

2.5.1. Analysis on 'Maha'

After performing 'Maha' transformation on training data, we found overlapping of different classes in almost all dimensions. Figure 13 (left) shows an example of overlapping after performing 'Maha' transformation and K-means. There is an obvious overlapping of data in different classes. This leads to large clustering error for 'Maha1', and thus gives poor classification rate after clustering.

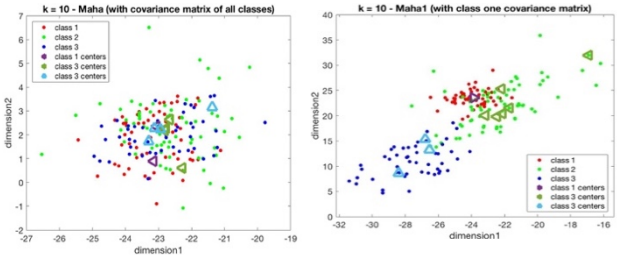

Figure 13 Data Distribution of 'Maha' (left) and 'Maha1' (right)

2.5.2. Analysis on 'Maha1'

Figure 13 (right) shows the distribution of training data after performing 'Maha1' transformation. We didn't find much overlapping at all dimensions. Instead, there are some outliers after 'Maha1' transformation, which explains the large error rate for 'Maha1' when K=3. Thus, we concluded that having more clusters solves the problem of misclustering due to outliers in some classes, such as Class 2 shown in the Figure.

2.6. Comparison of Results in Section 1 & 2

In section 1, different distance metrics were applied directly to exhaustively search for the training data that has the smallest distance to the test data, and then assigned the label of the training data to the test data to compute the error rate. Comparing to the results from section 2 (when K=3 and K=10), this method gives smaller error rates for all distance metrics applied in this coursework.

However, we consider that the method we used for classification in section 2 is a good approximation to that used in section 1, especially when the value of K is large, e.g. K=10. The advantage is that each test data only needs to search for K number of clusters to find a nearest one. Thus, if computational time is a major concern, the method we used in section 2 with appropriate value of K would give similar error rate while reducing the amount of computation time to a great degree. In addition, the choice of K depends on the data distribution as discussed in 2.5.1 & 2.5.2.

3.Neural Network

3.1 Construction and Training of Neural Network

Every neural network has an input layer, an output layer and a hidden layer. Input and output layers are determined by the dimension of input data and number of categories respectively. However, the hidden layer can vary greatly from one network to another as there can be any number of hidden layers. Too few layers can lead to slow learning process and too many hidden layers could result in overfitting. Furthermore, each hidden layer is independent and can have any number of hidden neurons. Every hidden neuron takes in all the output from each neuron in the previous layer and generates an output according to the activation function.

Training of the neural network is the process of minimising the error between input and output data set. The error is backward propagated through each layer, and weightings of neurons are adjusted to decrease the error. After several cycles the error will stop to decrease and training will be completed. The training function determines how the error is reduced.

Consequently, neural networks can differ in structure and complexity significantly. We will consider the main controllable inputs and compare the performance of different configurations based on classification error. To train a neural network, data also needs to be split into training, validating, and testing data. Different data partitioning can have a great impact on the classification error. We will begin by testing the same data partition as used in previous sections and compare the results, and then move on to a random partition to compare the performance of different partitions.

3.2 Analysis of Original Partition

3.2.1 Comparison of Activation Functions

We will begin by analysing the case with one hidden layer. There are three activation functions, log-sigmoid, tan-sigmoid and linear, where their shapes are depicted in Figure 14. The output of log-sigmoid function is between 0 and 1, whereas the tan-sigmoid output is between -1 and 1 and the output of linear function is the same as the input. All functions take in an input range of $-\infty$ to $+\infty$.

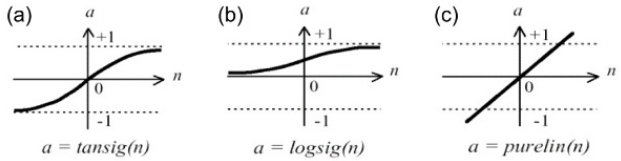

Figure 14 Different Activation Functions

The test is conducted with all three activation functions and the results are shown below in Figure 15 below, where each graph is labelled with the configuration that kept error within 2%. There are 17 training functions that are tested, all with 1 to 20 hidden neurons with one hidden layer and classification error is averaged over 5 tests. All three figures suggest there is a strong correlation between the type of training function and average error. Some training function perform poorly regardless of how many hidden neurons are used but others show a better average performance. For that reason, we will neglect the training functions which performed poorly on a consistent basis in the following sections.

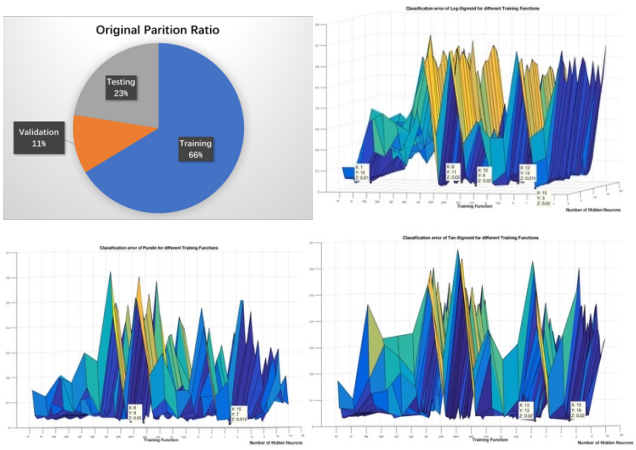

Figure 15 Classification Error for Different Training Functions and Activation Functions using Original Data Partition. Partition Ratio (top left), Log-Sigmoid (top right), Purelin (bottom left) and Tan-Sigmoid (bottom right)

3.2.2 Comparison of Training Functions

Regarding the training function, the ones achieved 2% or below error rate are Levenberg–Marquardt backpropagation (trainlm) training function, Gradient Descent with Adaptive Learning Rate (traingda) and Cyclical Order Weight/Bias training (trainc). The trainlm function belongs to the Quasi-Newton set, which is usually fast but with a disadvantage of memory and computation overhead. Traingda is one of the basic gradient decent algorithms but has additional implementation of adjustable learning rate which improves the performance.

The training time of trainlm and traingda for one cycle is 2.09 seconds and 2.38 seconds respectively. On the other hand, trainc requires 699.1 seconds thus we will not consider trainc for this problem.

The best performance of all is using trainlm with Log-Sigmoid activation function and 12 hidden neurons, which resulted in 1% classification error.

3.2.3 Selection of Hidden Layers

By adding a second hidden layer, the network becomes more complex. The test is conducted by adding one more layer to the previous best structure, and adjusting the hidden neurons on the second layer from 1 to 15 and the average result over 5 tests is depicted in Figure 16 below. We can see that the new network achieved a minimum of 4.5% with 9 neurons and failed to improve performance. This is because for this set of data, 2 hidden layers is too complicated and causes overfitting. Since adding more layers to the network will increase overfitting, we conclude that one hidden layer is optimal for this problem.

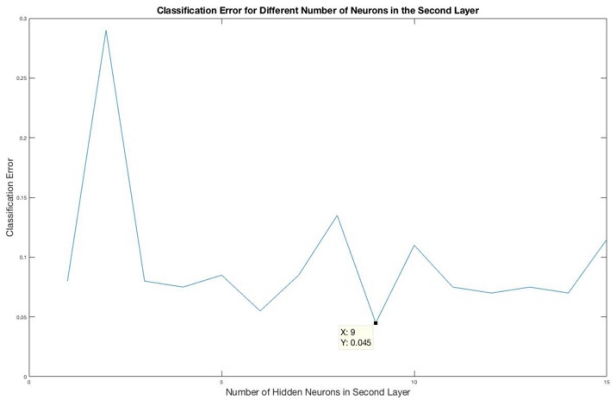

Figure 16 Classification Error with Two Hidden Layers

3.2.4 Comparison with results in previous sections

As mentioned in the section 2, NN classification after K-means is a good approximation for NN classification over all training data but with a significant reduction in computational time. Neural networks would provide a much better accuracy than both methods above, but at an expense of a substantial increase in computation time. This is because there is no systematic way to determine the best parameters to use for each problem, thus an exhaustive search would be needed. For this problem, as the data set contains only 13 dimensions, neural network could not show significant advantage in performance. However, for more complex patterns, neural networks will have more potential and provide lower error rates. Furthermore, it should be noted that K-means algorithm is unsupervised learning algorithm, as it does not need any training class labels, thus in some cases where there is lack of input data, this would be the only option.

3.3 Analysis of Random Partition

Now, we repeat the experiment with another random partition of train, validation and test data of 23%, 55% and 22% respectively, and the results are shown in Figure 17. The original data partition is displayed in white and the new partition is shown in colour. We can see clearly that with all three activation functions, the coloured surface is enclosed by the white surface from below. Therefore, the new random partition generally has a higher classification error than the original. The reason for this is that with such a small proportion of training data, the samples cannot represent the general distribution of the classes as good as the previous case. Even with a large percentage of data allocated for fine tuning, the general fitting cannot be improved much further.

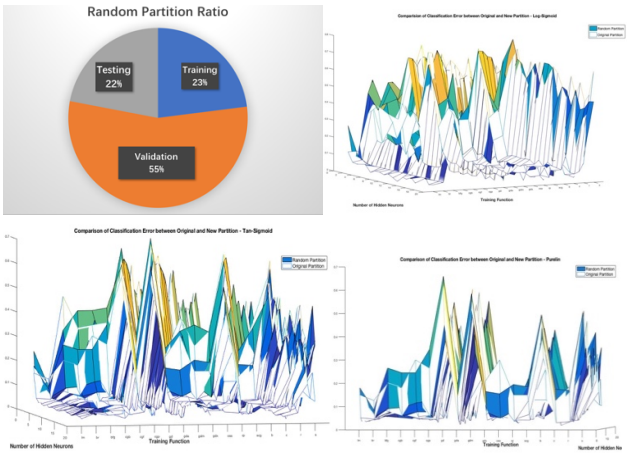

Figure 17 Comparison of Classification Error between Original and New Partition. Partition ratio (top left), Log-Sigmoid (top right), Tan-Sigmoid (bottom left) and Purelin (bottom right)