Vote Counting Measures for Ensemble Classifiers

Terry Windeatt
Centre for Vision, Speech and Signal Processing
University of Surrey, Guildford, Surrey, UK, GU2 7XH
Email: t.windeatt@surrey.ac.uk, Phone: 044-01483-689286 Fax: 044-01483-686031

Extended Abstract

Understanding the effectiveness of the Multiple Classifier Systems (MCS) framework has prompted the definition of various vote counting measures. The Margin concept was used originally to help explain Boosting. Bias and Variance are concepts from regression theory that have motivated modified definitions for 0/1 loss function for characterising Bagging and other ensemble techniques. Although it is known that diversity among base classifiers is a necessary condition for improved performance, there is no general agreement about how to quantify the notion of diversity among a set of classifiers. Various diversity measures have been studied with the intention of determining whether they correlate with ensemble accuracy. However, it is not clear or straightforward how to use the information available from any of these measures to assist in MCS design. The most common design approach is to rely on either a validation set or cross-validation techniques to select MCS parameter values.

In this paper, we propose a spectral representation of a Boolean function for analysing the feature space defined by a set of binary base classifiers. In contrast to the conventional method of representing a Boolean function as single vertices of the binary hypercube, a spectral representation incorporates global information about the function in each spectral coefficient. We assume that spectral techniques are more likely to be useful for realistic Pattern Recognition applications dealing with noisy data. Since the data is also incompletely specified and may be contradictory, we concentrate upon information content rather than computing the inverse transform. For the purposes of this paper, we exploit known properties of the spectral representation with respect to its ability to characterise high order correlations in the data. In MCS, the function is defined by the mapping from the set of binary classifiers to the target labels. By computing the first order spectral coefficients, we show that it is possible to define a measure of separability for the collection of half-spaces implied by the classifiers. Each pair of patterns, drawn from different classes, is rated according to contribution to separability. In our experiments, separability of a dataset is measured as the base classifier complexity is varied.

For the experiments reported here, a set of single hidden-layer MLPs connected in parallel serve as base classifiers. The training strategy for each individual MLP requires the setting of a number of parameters, which is considered by many to be the main drawback of these powerful classifiers. In our experiments, two MLP training parameters (numbers of hidden nodes and epochs) are systematically varied to simulate variation in base classifier complexity. Experimental evidence suggests that the proposed measure is correlated with base classifier test error as base classifier complexity is varied. The conclusion is that further investigation is warranted into the use of spectral techniques for MCS.

Vote Counting Measures for Ensemble Classifiers

Terry Windeatt
Centre for Vision, Speech and Signal Processing
University of Surrey, Guildford, Surrey, UK, GU2 7XH
Email: t.windeatt@surrey.ac.uk, Phone: 044-01483-689286 Fax: 044-01483-686031

Abstract
Various measures, such as Margin and Bias/Variance, have been proposed with the aim of gaining a better understanding of why Multiple Classifier Systems (MCS) perform as well as they do. While these measures provide different perspectives for MCS analysis, it is not clear how to use them for MCS design. In this paper a different measure based on a spectral representation is proposed for two-class problems. It incorporates terms representing positive and negative correlation of pairs of training patterns with respect to class labels. Experiments employing MLP base classifiers, in which parameters are fixed but systematically varied, demonstrate the sensitivity of the proposed measure to base classifier complexity.

Keywords: decision level fusion, multiple classifiers, ensembles, error-correcting, binary coding

1. Introduction

Traditionally, the approach used in the design of pattern classification systems has been to experimentally compare the performance of several classifiers in order to select the best one. However, an alternative approach based on combining multiple classifiers, has emerged over recent years and represents a departure from the traditional strategy. This approach goes under various names such as Multiple Classifier Systems (MCS) or committee or ensemble of classifiers, and has been developed to address the practical problem of designing automatic pattern recognition systems with improved accuracy. Recognising that each classifier may make different and perhaps complementary errors, the aim is to design a composite system that outperforms any individual classifier by pooling together the decisions of all classifiers. The expected result is that a single complex classifier may be advantageously replaced by a combination of relatively simple classifiers, often called base classifiers in the ensemble context.

It is clear that if all classifiers are identical there can be no advantage in combining their decisions, and therefore some differences between base classifiers is a necessary condition for improvement. Indeed, some MCS techniques actively attempt to perturb some aspect of the training set, such as features, patterns or

labels in order to force classifier diversity [1]. The best-known perturbation techniques are Bagging and Boosting, both of which have been extensively studied. Bagging [2] and the original form of Boosting [3] are both voting algorithms. Bagging (from Bootstrap Aggregating) forms replicate training sets by sampling with replacement, and combines the resultant classifications with a majority vote. Boosting, which combines with a weighted vote is more complex than Bagging in that the distribution of the training set is adaptively changed based upon the performance of sequentially constructed classifiers.

Understanding the effectiveness of the MCS framework has received much attention in the literature. The Margin concept (section 3.1) was used originally to help explain Boosting. Bias and Variance (section 3.2) are concepts from regression theory that have motivated modified definitions for 0/1 loss function for characterising Bagging and other ensemble techniques. Different but related approaches that have contributed to a theoretical understanding of MCS include [4] and [5]. Although it is known that diversity among base classifiers is a necessary condition for improved performance, there is no general agreement about how to quantify the notion of diversity among a set of classifiers. As discussed in Section 3.3, various diversity measures have been studied with the intention of determining whether they correlate with ensemble accuracy. However, it is not clear or straightforward how to use the information available from any of these measures to assist in MCS design. The most common design approach is to rely on either a validation set or cross-validation techniques to select MCS parameter values.

In this paper, we propose a spectral representation of a Boolean function for analysing the feature space defined by a set of binary base classifiers. In contrast to the conventional method of representing a Boolean function as single vertices of the binary hypercube, a spectral representation incorporates global information about the function in each spectral coefficient. We assume that spectral techniques are more likely to be useful for realistic Pattern Recognition applications dealing with noisy data. Since the data is also incompletely specified and may be contradictory, we concentrate upon information content rather than computing the inverse transform. Therefore, it is not too important which spectral ordering we choose, but we will refer to the Rademacher-Walsh spectrum. The Rademacher functions [6] are an incomplete set of

orthogonal functions that can generate the complete set of Walsh functions, though not in original Walsh order. This alternative order, referred to as Rademacher-Walsh, is the one most frequently used for Threshold Logic Unit (TLU) synthesis, as discussed in [7] [8] using spectral decomposition, summation and translation.

For the purposes of this paper, we concentrate on known properties of the spectral representation with respect to its ability to characterise high order correlations in the data (section 0). In MCS, the function is defined by the mapping from the set of binary classifiers to the target labels. By estimating the first order spectral coefficients, we show that it is possible to define a measure that characterises the collection of half-spaces implied by the classifiers. Each pair of patterns, drawn from different classes, is rated according to contribution to separability. In our experiments, the proposed measure is shown to correlate well with base classifier test error suggesting the possibility of selecting base classifier parameters based on information available from the training set.

We know that model selection using training data alone requires a built-in assumption, since realistic learning problems are in general ill-posed [9]. Ways of building in appropriate assumptions have been investigated in various contexts, for example in certain decision tree pruning methods [10] and information-theoretic approaches [11] [12] [13]. Here, the assumption is built into the estimation of spectral coefficients, namely that the spectral contribution with respect to a pair of patterns is inversely proportional to Hamming Distance (Equation (2), section 2.3). Hamming Distance was also used as a measure of closeness between binary patterns in the decision-making stage of Error-Correcting Output Coding (ECOC) [14] [15]. The principle behind ECOC is that individual classification errors can be tolerated if codes with large Hamming distance are employed. In ECOC however, L_1 *norm* usually replaces Hamming Distance if it can be shown that base classifiers provide good probability estimates [16]).

For the experiments reported here, a set of single hidden-layer MLPs connected in parallel serve as base classifiers. The training strategy for each individual MLP requires the setting of a number of parameters, which is considered by many to be the main drawback of these powerful classifiers. In our experiments, two MLP training parameters (numbers of hidden nodes and epochs) are systematically varied to simulate variation in base classifier complexity. Training set perturbation methods were generally developed with classification trees as base classifiers, and do not necessarily improve performance with neural network base classifiers. In this paper we do not perturb the training set as in Bagging, but use MLP weight initialisation, which provides inherent random perturbation. The ability to vary complexity in a smooth way is not shared by decision trees and a neural network is an appropriate choice if we want to determine sensitivity of the proposed measure to base classifier complexity.

The paper is organised as follows. The meaning and calculation of spectral coefficients, along with the definition of the proposed measure is given in section 2. Other vote counting measures including Margin and Bias/Variance are discussed in Section 3, with experimental evidence using artificial and real data presented in Section 4. An Appendix that explains the underlying concepts and relationship to separability is provided after the Conclusion.

2. Spectral Representation

This section is split into four topics. The first topic in section 0 deals with the spectral representation of a Boolean function *f(X)* and reviews established theory pertaining to the meaning of spectral coefficients [8]. Secondly, in Section 2.2 we propose a representation of *f(X)* that enables spectral coefficients to be easily computed. The advantage of this representation is that, with suitable assumptions, it enables the estimation of spectral coefficients even when *f(X* is noisy, incomplete and contradictory, as discussed in section 2.3. The final topic deals with the definition of the proposed measure in section 2.4.

Assume that we are dealing with a two-class problem and that one of two classes is assigned to each of *b* base classifiers, so that the *mth* training pattern X_m may be represented as a vertex in the *b*-dimensional binary hypercube

$$X_m = (x_{m1}, x_{m2}, \ldots, x_{mb}) \qquad x_{mj} \text{ and } f(X_m) \in \{+1,-1\} \qquad (1)$$

The following equations assume *{+1,-1}* coding and a simple modification is required for *{0,1}* coding

2.1 Spectral coefficients and meaning

The transformation of binary data can be carried out using a variety of matrices $\begin{bmatrix} T^{n-1} & T^{n-1} \\ T^{n-1} & -T^{n-1} \end{bmatrix}$ that differ only in row ordering. The Hadamard transform T^n with entries $\in$ {+1,–1} is a complete orthogonal square matrix that can be expressed as a recursive structure:

The Walsh and Rademacher-Walsh transform matrices have similar row entries but use a different ordering of the 2^n functions that collectively constitute the closed set. The inverse for all these three orderings exists and is given by $(2^n)^{-1}[T^n]^t$. However, since the functions we deal with are incompletely specified and noisy we do not attempt to compute the inverse transform, which would necessitate {+1, -1} coding. Therefore, we can use any spectral ordering and any binary coding for both features and target. Assuming that the transform is represented by $T^nY = S$, in [8] the subscript notation and corresponding meaning for coefficients up to third order is given as follows:

s_0	correlation between *f(X)* and constant	
s_i *i=1...n*	correlation between *f(X)* and x_i	(2)
s_{ij} *i,j = 1...n, i≠j*	correlation between *f(X)* and $x_i \oplus x_j$	
s_{ijk} *i,j,k = 1...n, i≠j≠k*	correlation between *f(X)* and $x_i \oplus x_j \oplus x_k$	
..... and continues for fourth order and above		

where ⊕ is logic exclusive-OR. All higher order coefficients can be computed from a simple summation of first order contributions [17] (see example 1). Interestingly, it is known that first order coefficients, $s_{i,}$

provide a unique identifier of *f(X)* if it is linearly separable. Although there is no known mathematical relationship between s_i and weight/threshold values of a Threshold Logic Unit (TLU) implementation, tables exist for $n \leq 7$ (for example see [8]).

2.2 Spectral coefficient calculation

The proposed representation of *f(X)* is based on the concept of sensitivity σ and its motivation is explained in the Appendix. Informally, σ indicates whether a change in binary value x_j gives rise to a change in *f(X).* The information is implicit in the original representation, but by making it explicit, excitatory and inhibitory spectral contributions can be easily computed [18]. For a completely specified function in {+1,1} coding the *mth* pattern component x_{mj} is assigned σ_{mj} ($j=1,2,...b$) as follows

$$\sigma_{mj} = \frac{\mathbf{1} - \boldsymbol{x}_{mj}\boldsymbol{x}_{nj}}{\mathbf{2}}, \quad \boldsymbol{f}(\boldsymbol{X}_m) \neq \boldsymbol{f}(\boldsymbol{X}_n), \quad \sum_{j=1}^{b}\left(\frac{\mathbf{1} - \boldsymbol{x}_{mj}\boldsymbol{x}_{nj}}{\mathbf{2}}\right) = \mathbf{1} \qquad (3)$$

where Hamming Distance $\boldsymbol{D}_{\boldsymbol{H}}(\boldsymbol{X}_m, \boldsymbol{X}_n) = \sum_{j=1}^{b}\left(\frac{\mathbf{1} - \boldsymbol{x}_{mj}\boldsymbol{x}_{nj}}{\mathbf{2}}\right)$

In {0,1} coding this changes to

$$\sigma_{mj} = \left|\boldsymbol{x}_{mj} \oplus \boldsymbol{x}_{nj}\right|, \quad \boldsymbol{f}(\boldsymbol{X}_m) \neq \boldsymbol{f}(\boldsymbol{X}_n), \quad \sum_{j=1}^{b}\left|\boldsymbol{x}_{mj} \oplus \boldsymbol{x}_{nj}\right| = \mathbf{1}$$

In order to keep excitatory and inhibitory contributions separate, σ_{mj} is defined as

$$\begin{cases} \text{excitatory or positive correlation, denoted } \sigma_{mj}^{+} & \text{if } x_{mj} = f(X_m) \\ \text{inhibitory or negative correlation, denoted } \sigma_{mj}^{-} & \text{if } x_{mj} \neq f(X_m) \end{cases}$$

After applying equation (3), each pattern component x_j has associated σ_j which is written as $x_j^{\sigma j}$. Using spectral summation [8] the difference between excitatory and inhibitory contributions, $\sum_X \sigma_j^{+}$ and $\sum_X \sigma_j^{-}$ gives the first order spectral coefficient s_j. The existence of $\sum_X \sigma_j^{+} > 0$ and $\sum_X \sigma_j^{-} > 0$ for

given *j* provides evidence that the set of patterns is not *1*-monotonic in the *j*th component and therefore non-separable (1-monotonic check (a1) in the Appendix).

Example 1 demonstrates two ways of computing the spectral coefficients of *f(X).*

Example 1: Spectral coefficient calculation: non-separable function

$$f(X) = (\bar{x}_1 \cap x_2) \cup (x_1 \cap \bar{x}_2) \cup (x_2 \cap x_3)$$

With reference to the meaning of coefficients given in (2), the spectral ordering associated with matrix multiplication given in Figure 1 assumes the truth table ordering of Table 1. An alternative calculation of the spectral coefficients is obtained by applying equation (3), as shown in Figure 2. The three rows in Figure 2 represent class 1 binary patterns, and s_1, s_2, s_3 are the first order spectral coefficients. Only class 1 is shown and, by duality, there is an identical contribution from class -1. To calculate higher order coefficients, the first order contributions are added for the respective columns, ignoring any component with $\sigma_j = 0$.

e.g. s_1 = 2 * (1-1+1) = +2, using column 1.
s_{12} = 2 * ((1 * 1) + (-1 * -1) + (1 * 1)) = +6 using column 1,2.
s_{123} = 0 + 2 * (-1 * -1 * 1) + 0 = +2 using column 1,2,3.

$\sum_X \sigma_j^+ / \sum_X \sigma_j^-$ *(j = 1,2,3)* = [4/2,4/2,2/0] , showing that the function is not 1-monotonic in the first two components from (a1) in the Appendix, and is therefore non-separable.

2.3 Spectral coefficient estimation for Incompletely specified functions

For noisy incompletely specified and perhaps contradictory patterns, equation (3) needs to be modified.

$$\sigma_{mj} = \sum_{Xn} \left(\frac{1 - x_{mj} x_{nj}}{2 D_H (X_m, X_n)} \right), \qquad f(X_m) \neq f(X_n), \tag{4}$$

where σ_{mj} is denoted σ_{mj}^+ *and* σ_{mj}^- as defined in Rule 1

In contrast to equation (3), in equation (4) all pattern pairs drawn from different classes are considered, not just nearest D_H neighbours. With no evidence to the contrary, the contribution to a pattern pair is assumed to be inversely proportional to H_D and shared equally between all pattern components that differ.

2.4 Spectral coefficients and Separability

In the MCS representation given in (1), the jth component x_j of a pattern pair has associated σ_j^- after applying equation (4) only if the jth base classifier mis-classifies both patterns. Therefore we expect that a pattern with relatively large $\sum_{j=1}^{b} \sigma_j^-$ is likely to come from regions where the two classes overlap. In order to characterise a pattern's contribution to separability we look at the relative contribution with respect to $\sum_X \sigma_j^+$ and $\sum_X \sigma_j^-$. The proposed measure σ_T for a pattern is defined to be the difference between relative excitatory and inhibitory contributions, normalised so that $-1 \le \sigma_T \le 1$.

$$\sigma_T = \frac{1}{N} \text{ x } \sum_{j=1}^{b} \left[\left(\frac{\sigma_j^+}{\sum_X \sigma_j^+} - \frac{\sigma_j^-}{\sum_X \sigma_j^-} \right) \right] \qquad (5)$$

$$\text{where } N = \sum_{j=1}^{b} \left[\left(\frac{\sigma_j^+}{\sum_X \sigma_j^+} + \frac{\sigma_j^-}{\sum_X \sigma_j^-} \right) \right] \quad \text{and} \quad \sum_X \text{ is sum over all training patterns}$$

Cumulative distribution graphs for σ_T are defined similar to cumulative distribution graphs for Margin (section 3.1), that is $f(\sigma_T)$ versus σ_T where $f(\sigma_T)$ is the fraction of patterns with separability at least σ_T.

Experimental results on natural benchmark datasets in section 4 show that, as base classifier complexity is increased, the σ_T distribution appears to be sensitive to the (training set) contribution from overlap regions.

3. Vote Counting measures

Various vote counting measures have been proposed with the aim of gaining a better understanding of MCS performance. Margin is simply found by counting the additional votes received by the winning class with respect to the second winner (section 3.1). Most Bias/Variance definitions are defined as a vote count comparison between base classifiers and Bayes classifier (section 3.2). Pair-wise diversity measures are functions of misclassification counts for a pair of classifiers (section 3.3). Note also that the McNemar significance test [19], which we use in section 4, applies the chi test which involves thresholding a function of the counts defined in section 3.3.

3.1 Margin

Margin was originally developed for Support Vector Machines and Boosting. In [20]the margin concept was used to analyse Boosting and thereby understand its effectiveness in generalising well even though the training error drops exponentially fast. The intention was to explain why, for many problems, Adaboost appears not to over-fit with increasing number of classifiers despite the training error reducing to zero. The Margin of a training example is defined as the difference between the weight given to the correct class and the maximum weight given to any of the other classes. It is defined as a number between –1 and +1, and is positive for a correct classification. Furthermore, the absolute value of the Margin represents confidence of classification. For a two-class problem, the Margin for *mth* training pattern X_m with target ω_m is given by

$$\textbf{Margin}(X_m, f(X_m)) = \frac{f(X_m)\sum_{j=1}^{b}\alpha_j x_{mj}}{\sum_{j=1}^{b}\left|\alpha_j\right|} \qquad (6)$$

where α_j is the weight associated with *jth* base classifier

Note that Margin for majority vote ($\alpha_j = 1/b$) is identical to unnormalised s_0 defined in (2), so that Margin may be regarded as a special case of spectral summation.

It is customary to plot Margins as cumulative distribution graphs, that is f (z) versus z where z is the Margin and f (z) is the fraction of patterns with Margin at least z. A Margin distribution curve that moves *to*

the right is indicative of a more confident classification as given in [20], and in which it is also proved that larger Margins are associated with superior upper bounds on the generalisation error. It is also shown that the derived bound is independent of the number of classifiers. However, as pointed out by the authors, the bounds are not necessarily tight and therefore of limited practical usefulness. Larger Margins are sometimes consistent with higher prediction error, and Adaboost is known to over-fit on some high noise problems even though Margins are increased. The problem appears to be that difficult patterns, such as outliers and misclassifications, can distort the Margin distribution. In [21] it is claimed that tighter and therefore more useful bounds are derived. In [22] a distinction is made between hard and soft Margin similar to SVM Margin optimisation, and a soft Margin is proposed that allows the possibility of mistrusting patterns associated with repeated misclassifications.

3.2 Bias/Variance

The use of Bias and Variance for analysing multiple classifiers is motivated by what appears to be analogous concepts in regression theory. It might be supposed that averaging a large number of classifiers leads to a smoothing out of the error rates. Indeed, visualisation of simple two-dimensional problems appears to support the idea that Bias/Variance is a good way of quantifying the difference between the Bayes decision boundary and the combined classifier boundary. However there are at least three reported fundamental difficulties with the various Bias/Variance definitions for 0/1 loss functions.

First, a comparison of Bias/Variance definitions [23] shows that no definition satisfies all properties that would ideally be expected for 0/1 loss function. In particular, it is shown that it is impossible for a single definition to satisfy both:

1) zero Bias and Variance for Bayes classifier
2) additive Bias and Variance decomposition of error (as in regression theory)

Secondly, as the authors of the various Bias/Variance definitions state, the effect of bias and variance on error rate cannot be guaranteed. It is easy to think of example probability distributions for which bias and variance are constant but error rate changes with distribution, or for which reduction in variance leads to increase in error rate [23] [25]. Besides these two theoretical difficulties, there is the additional

consideration that for real problems the Bayes classification needs to be known or estimated. Although some definitions, for example [24], do not require this, the consequence is that the Bayes error is ignored.

In our experiments, we use Breiman's definition [25] which is based on defining Variance as the component of classification error that is eliminated by aggregation. Patterns are divided into two sets, the Bias set B containing patterns for which the Bayes classification disagrees with the aggregate classifier and the Unbias set U containing the remainder. Bias is computed using B patterns and Variance is computed using U patterns, but both Bias and Variance are defined as the difference between the probabilities that the Bayes and base classifier predict the correct class label. Therefore, the reducible error (what we have control over) with respect to a pattern is either assigned to Bias or Variance, an assumption that has been criticised [23]. However, this definition has the nice property that the error of the base classifiers can be decomposed into additive components of Bayes error, Bias and Variance.

3.3 Diversity Measures

Although it is recognised in classifier combination that a diverse ensemble has better potential for improved accuracy, the notion of diversity is not well defined. Various approaches to measuring diversity, and to determining the relationship between diversity and accuracy, have been proposed. Pair-wise measures depend on the following four counts *c1-c4*, and are defined with respect to the number of patterns for which a classifier pair (A, B):

c1:	A and B both correct	11|1 or 00|0	where ab|ω denotes by a, b the classifier decisions
c2:	A and B both misclassified	11|0 or 00|1	
c3:	A correct and B misclassified	10|1 or 01|0	
c4:	B correct and A misclassified	10|0 or 01|1	

The Double Fault Measure is just c4, and the Disagreement Measure is *c2 + c3*, while the Q statistic and Correlation Coefficient are more complex functions of these counts [26]. In order to apply pair-wise measures to finding overall diversity of a set of classifiers it is necessary to average over the set. Non-pair-

wise measures attempt to measure diversity of a set of classifiers directly, based on variance, entropy or based on proportion of classifiers that fail on randomly selected patterns. The main difficulty with diversity measures is the so-called accuracy-diversity dilemma. As explained in [26], as the classifiers approach the highest levels of accuracy, diversity must decrease. Therefore, it is expected that there will be a trade-off between diversity and accuracy, and an accuracy-diversity diagram is a useful way of visualising the relationship. However, there has been no convincing theory or experimental study to suggest that any of the measures provides a good predictor of generalisation error of an ensemble.

4. Experimental Evidence

Natural two-class benchmark problems have been selected from [27] and [28], and the experiments use random 50/50 or 20/80 training/testing splits. The artificial data is from [25], and uses 300 training patterns and 3000 test patterns.

All experiments are performed with one hundred single hidden-layer MLPs serving as base classifiers. To minimise the number of parameters to vary, we choose the Levenberg-Marquardt training algorithm with default parameters. While all the parameters of the base classifier MLPs are fixed at the same values, we systematically vary the numbers of hidden nodes and training epochs for different runs of the MCS. Unless otherwise specified, number of nodes varies from 2-16 [2,4,8,16] and number of epochs from 1-32 [1,2,4,8,16,32]. Each node-epoch combination is repeated twenty times for Diabetes and ten times for all other datasets. In the experiments described here, random perturbation is caused by different starting weights on each run. The architecture is simple, with parallel base classifiers and outputs combined by majority vote.

For Diabetes 50/50, we have produced a set of curves shown in Figure 4 to Figure 9. The Diabetes dataset is known to over-fit and perform poorly with Boosting and other methods [5] [22]. Figure 4 shows the majority vote and base classifier error rates for training and testing. Figure 4 (a) compared with Figure 4 (c) demonstrates that over-training of the majority vote classifier begins at 4 epochs for 4, 8 and 16 nodes. Similarly Figure 4 (b) and (d) show that over-fitting of the base classifier begins at 8 epochs for 8 and 16 nodes.

Margin distributions ($0 \leq$ Margin ≤ 1), defined in Section 3.1, are shown in Figure 5, but with 1 and 2 epoch curves not shown for clarity. It may be noted that the majority vote training error rate (Figure 4 (c)) can be derived directly from the zero Margin intercepts in Figure 5. From Figure 5, it can also be seen that as the number of epochs increases the curve moves to the right, which was interpreted in [20] as indicating more confident classification. The σ_T distributions ($0 \leq \sigma_T \leq 1$), are shown in Figure 6 and demonstrate that, for 16 and 32 epochs, 8 and 16 nodes the curve does not necessarily move to the right. To quantify this curve movement, Figure 7 (d) and Figure 7 (a) show the area under Margin and σ_T distributions. Similarly, Figure 7 (b) (c) show σ_T distribution plotted as area under curve for ($-0.1 \leq \sigma_T \leq 0.1$) and for ($-1 \leq \sigma_T \leq 0$). Figure 7 (b) represents a robust measure for the number of patterns with negative σ_T (used in [29] for partitioning the training set). Comparison of Figure 7 and Figure 4 (b) suggests that area under σ_T distributions may be correlated with base classifier test error rate.

Figure 8 shows Bias and Variance (Breiman definition section 3.2) calculated on the test set. Since we need to know the Bayes classification to compute Bias and Variance, we make the optimistic assumption that the lowest majority vote test error rate (for this problem 2 nodes at 8 epochs) corresponds to Bayes classification. Hence, the Bias in Figure 8 (a) is 0 percent at 2 nodes, 8 epochs. Furthermore the decomposition of Bias and Variance means that Figure 4 (b), the base classifier error rate, can be found by adding together the estimated Bayes rate (twenty-three percent), Figure 8 (a) and Figure 8 (b). (Decomposition is not exact due to our assumption for estimating Bayes classification).

To appreciate the significance of the results, Figure 9 shows the standard deviation of area under σ_T distributions and the number (%) of significant differences of majority vote (McNemar 5%) with respect to best majority vote error rate (2 nodes at 8 epochs).

In Figure 10 to **Error! Reference source not found.**, selected graphs are provided for Diabetes 20/80, Cancer 50/50 and 20/80. In each figure is shown, for varying number of epochs and nodes, the base

classifier and majority vote error on test set (a, b), test set Bias and Variance (c,d), and training set Margin and σ_T distributions (e, f). Similar curves were also produced for Ion 20/80, Heart 50/50 and 20/80, Vote 50/50, Credita 50/50., card 50/50 and 20/80, ringnorm, threenorm, twonorm. The 20/80 datasets generally showed similar performance to respective 50/50 datasets but with over-fitting occurring at lower number of epochs. In 50/50 and 20/80 datasets, the majority vote test error over-fitted at a lower number of epochs compared with base classifier error.

Results for Twonorm are shown in Figure 13. Bias and Variance calculations use the true Bayes classification, since it can be accurately determined by simulation The Bayes rate for the this problem is 2.3%.

The correlation coefficients between area under σ_T distribution ($0 \leq \sigma_T \leq 1$) and test errors (base classifier and majority vote) are given in Table 2, Table 3 and Table 4 for 50/50, 20/80 and artificial datasets respectively. To compare across different datasets, the correlation with respect to epochs is performed with the number of nodes set to the value that gave minimum majority vote test error rate. From Table 2, Table 3 and Table 4 it can be seen that area under σ_T distribution ($0 \leq \sigma_T \leq 1$) is well correlated with base classifier test error.

5. Conclusion

In this paper only two class classification problems are considered. A method for extending the technique to multi-class problems would be to incorporate the Output Coding method [30]. Indeed, the error-correcting principle behind Error-Correcting Output Coding is similar to the ideas proposed here [14].

Although computation of spectral coefficients for completely specified Boolean functions was studied over thirty years ago, the techniques need to be modified if they are to be applied to the MCS framework. We propose a representation that enables spectral coefficients to be separated into excitatory and inhibitory components, and that facilitates a simple assumption for handling incompletely specified, noisy and contradictory functions. We also define a pattern measure based on first order spectral contributions that is

intended to reflect the contribution of a pattern to the separability of a dataset. Experimental evidence suggests that the proposed measure is correlated with base classifier test error as number of training epochs is varied. The conclusion is that further investigation is warranted into the use of spectral techniques for MCS.

Appendix: Sensitivity, Separability and k-monotonicity

The following analysis of a feedforward network is based on the ideas of [31] and [32]. A thorough understanding requires some background in test generation of logical faults, explained in [33].

Consider the graph representation of a feedforward network of logic functions. We can assign binary values to inputs and propagate these values to all network lines by conventional logic simulation. Now assume that we want to determine the effects of changing the binary value of a single line in the network, which we call injecting a transition. To inject a binary $\bar{v}/v$ transition, on line λ having logic value $\bar{v}$ in a given network $\mathbf{N}$, we first copy $\mathbf{N}$ to $\mathbf{N}_{\oplus}$. We then make a cut at λ in $\mathbf{N}_{\oplus}$ and change the value from $\bar{v}$ *to* v, treating the point of cut as a pseudo-input thus pruning the branch and ignoring the subtree that feeds the branch. The changed value at λ in $\mathbf{N}_{\oplus}$ will usually lead to other changes in logic values between λ and network output. Binary transitions can be traced by specifying a suitable formalism that compares logic values of corresponding nodes in $\mathbf{N}$ and $\mathbf{N}_{\oplus}$.

Now the Boolean difference [33] of function $f(\xi_1, \xi_2 \ldots \xi_n)$ w.r.t. input ξ_i is given by

$$df(X)/d\xi_i = f(\xi_1,\ldots,\xi_i,\ldots\xi_n) \oplus f(\xi_1,\ldots, \bar{\xi}_i,\ldots\xi_n) \qquad (\oplus = \text{xor})$$

From $df(X)/d\xi_i$ we can find all input patterns for which f changes value in response to binary transition on ξ_i, given by the union of two sets

$$\{X : \xi_i\, df(X)/d\xi_i = 1\} \cup \{X : \bar{\xi}_i\, df(X)/d\xi_i = 1\}$$

In fact, this union holds for hidden node *h* if ξ_i is replaced by *h(X)* and *f(X)* by *f(X,h)*. Note that by deriving network $\mathtt{N}_{\oplus}$ from N we facilitate computation of $df(X)/d\xi_i$ without unwieldy manipulations normally associated with rules of Boolean Difference algebra.

We call a logic value $\bar{v}$ sensitive (insensitive) iff the injected $\bar{v}/v$ transition is (is not) propagated to network output. To each logic value (ξ) we attach another binary value that indicates sensitivity (σ). For *f(X)* each input ξ_j *(j = 1,...,n)* has associated sensitivity σ_j and for convenience, we write the jth component as $\xi_j^{\sigma_j}$. We can find $\xi_j^{\sigma_j}$ from a truth table by comparing input patterns that are unit Hamming Distance (H_D) apart, and setting $\sigma_j = 1$ if target response differs and $\sigma_j = 0$ otherwise (as specified in equation (3), Section 2.2). Note that for a completely specified Boolean function (all 2^b rows of the truth table) this rule for finding σ_j, for all *j* and over all patterns represents the first stage of logic minimisation such as Quine-McCluskey tabular method [34], and is identical in terms of complexity.

A discussion of *k*-monotonicity as necessary and increasingly sufficient conditions for separability is given in [35]. To understand monotonicity of a Boolean function, consider the following definitions pertaining to *n*-dimensional functions *f(X)* and *g(X)* adapted from [35]:

> *f* implies *g*, $f \subseteq g$ if any *X* satisfying *f(X)=1* also satisfies *g(X)=1*, but not necessarily conversely.
>
> *f* is *k*-comparable if either $f_A \subseteq f_{\bar{A}}$ or $f_A \supseteq f_{\bar{A}}$ holds for each *k*-assignment *A*, where *k*-assignment is an assignment of binary values to *k* out of *n* variables
>
> *f* is *m*-monotonic if *f* is *k*-comparable for every *k* such that $1 \le k \le m$. If *m = 1*, *f* is unate. If *m = n*, *f* is completely monotonic, a necessary but not sufficient(unless $n \le 8$) condition for linear separability.
>
> *f* is *m*-summable for a given *m*, if for some *k*, such that $2 \le k \le m$, there exist two sets $\{\mathbf{a}^{(j)} | f(\mathbf{a}^{(j)}) = 1\}$ and $\{\mathbf{b}^{(j)} | f(\mathbf{b}^{(j)}) = 0\}$, such that the vector summation (repetition allowed) $\sum_{j=1}^{k} \mathbf{a}^{(j)} = \sum_{j=1}^{k} \mathbf{b}^{(j)}$ holds.

f is *m*-asummable if *f* is not *m*-summable. If *f* is not m-summable for any $m \geq 2$, *f* is asummable, a sufficient and necessary condition for separability

This classification shows that the linearly separable class constitutes the most restrictive of a number of classes of Boolean functions. Between this and the unrestricted class of 2^{2^n} Boolean functions, are the *1*-monotonic, *2*-monotonic, higher monotonic, completely monotonic (2-asummable) [8]. Thus, there are degrees of non-separability for which appropriate checks provide necessary and increasingly restrictive conditions for a data set to be separable. If *f(X)* is linearly separable and *n*-dimensional, it can be implemented by a single TLU with weights w_i *(i = 1 ... n).*

Necessary and sufficient checks on *f(X)* for *k*-monotonicity, $k = 1$ and $k \geq 2$ are as follows:

1-monotonic check:

If any two patterns X_p, X_q can be found such that for the *i*th component:

$$\xi_{pi} = \bar{\xi}_{qi}, \ \sigma_{pi} = \sigma_{qi} = 1 \qquad \text{(a1)}$$

f(X) is not *1*-monotonic since *1*-assignment $A = \{\xi_{pi} \rightarrow 1\}$ shows it is not *1*-comparable. It implies that it is not possible to implement with a single TLU since the requirement is for weight w_i to be both excitatory and inhibitory.

2-monotonic check.

If a single pattern X_p can be found such that for *i*th and *j*th components:

$$\xi_i = \xi_j = f(X_p), \ \sigma_i = 1, \ \sigma_j = 0 \qquad \text{(a2)}$$

then $f_A \supset f_{\bar{A}}$ and the 2-assignment $A = \{\xi_{pi} \rightarrow 1, \xi_{pj} \rightarrow 0\}$ implies $w_i > w_j$. A similar check for $w_i < w_j$ exists if $\xi_i = \xi_j = \bar{f}(X_p)$,

k-monotonic check, $k \geq 2$

Since the weight constraint relationship is transitive [35] we can use weight ordering to check *k*-monotonicity, $k \geq 2$**.**

Examples A1 and A2 show, for two separable functions, how to count spectral contributions and check *1*-monotonicity (see [18] for examples of non-separable functions that are checked for complete monotonicity and implemented using layers of TLUs).

Example A1: Two-input NAND $f(X) = \bar{\xi}_1 + \bar{\xi}_2$

1	2	Target
1^0	1^0	-1
-1^0	1^1	-1
1^1	-1^0	-1
-1^1	-1^1	1

The table shows the result of applying equation (3). Note that a 1/-1 or -1/1 transition on one input is propagated to the output if and only if the other input is -1.

$\sum_X \sigma_j^+ / \sum_X \sigma_j^- \{j = 1,2\} =$ [0/2,0/2]. The function is *1*-monotonic from (a1).

Example A2: A separable Boolean function

$$f(X) = \xi_1\xi_2 + \xi_1\xi_3 + \xi_2\xi_3\xi_4$$

By applying equation (3) to the truth table, or by looking at Karnaugh map representation in Figure 3 :

$\sum_X \sigma_j^+ / \sum_X \sigma_j^- \quad (j = 1,2,3,4) =$ [10/0,6/0,6/0,2/0]. The function is *1*-monotonic from (a1).

T^n								Y		S	
1	1	1	1	1	1	1	1	1		-2	s_0
1	-1	1	-1	1	-1	1	-1	-1		+2	s_1
1	1	-1	-1	1	1	-1	-1	-1		+2	s_2
1	-1	-1	1	1	-1	-1	1	1	=	+6	s_{12}
1	1	1	1	-1	-1	-1	-1	1		+2	s_3
1	-1	1	-1	-1	1	-1	1	-1		-2	s_{13}
1	1	-1	-1	-1	-1	1	1	-1		-2	s_{23}
1	-1	-1	1	-1	1	1	-1	-1		+2	s_{123}

Figure 1: Matrix multiplication to calculate spectral coefficients in Example 1

$$\begin{matrix} s_1 & s_2 & s_3 \end{matrix}$$
$$\begin{bmatrix} +\mathbf{1}^{1} & +\mathbf{1}^{1} & +\mathbf{1}^{0} \\ -\mathbf{1}^{1} & -\mathbf{1}^{1} & +\mathbf{1}^{1} \\ +\mathbf{1}^{1} & +\mathbf{1}^{1} & -\mathbf{1}^{0} \end{bmatrix}$$

Figure 2: Class 1 patterns in {+1,-1} coding after applying equation (3) in Example 1

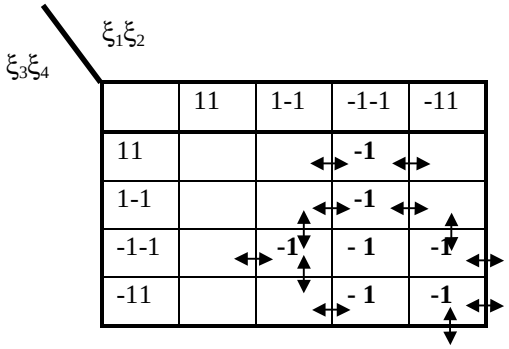

Figure 3: Karnaugh Map representation for Example A2 showing σ as double-headed arrows

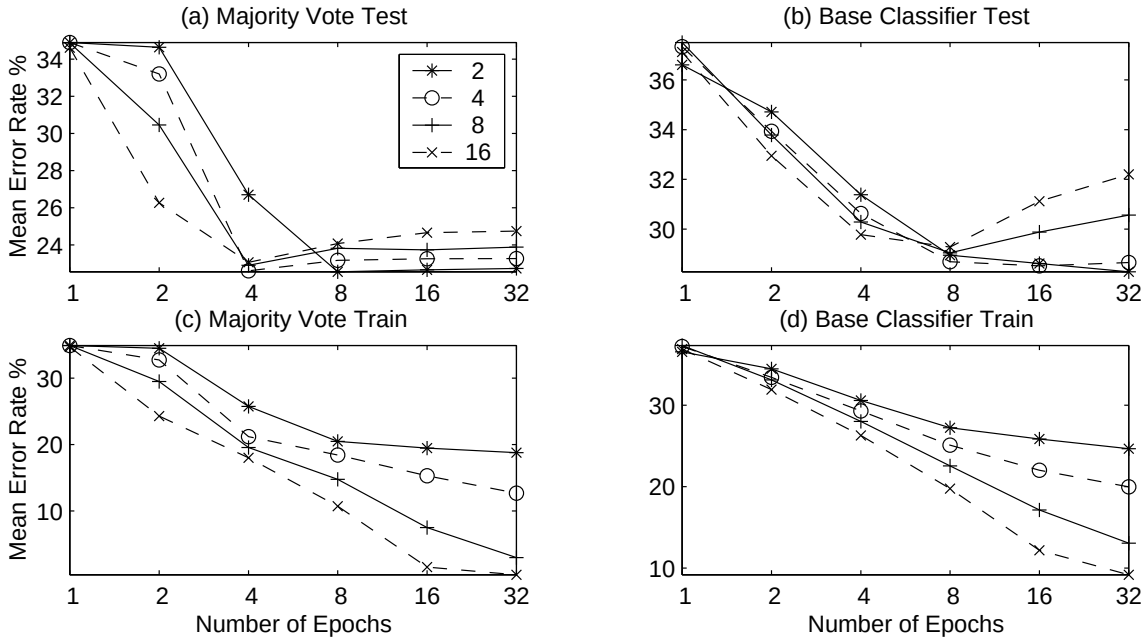

Figure 4: Base classifier and Majority Vote training and testing error rates, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

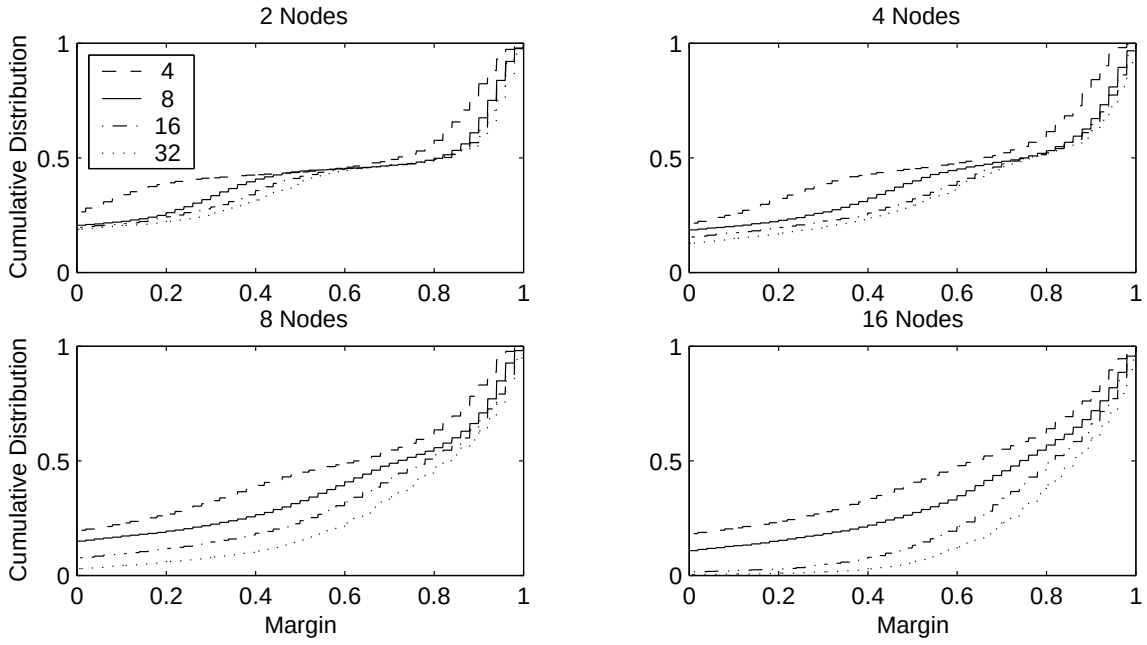

Figure 5: Margin distributions (0 ≤ Margin ≤ 1), Diabetes 50/50 for 4-32 epochs, 2-16 nodes

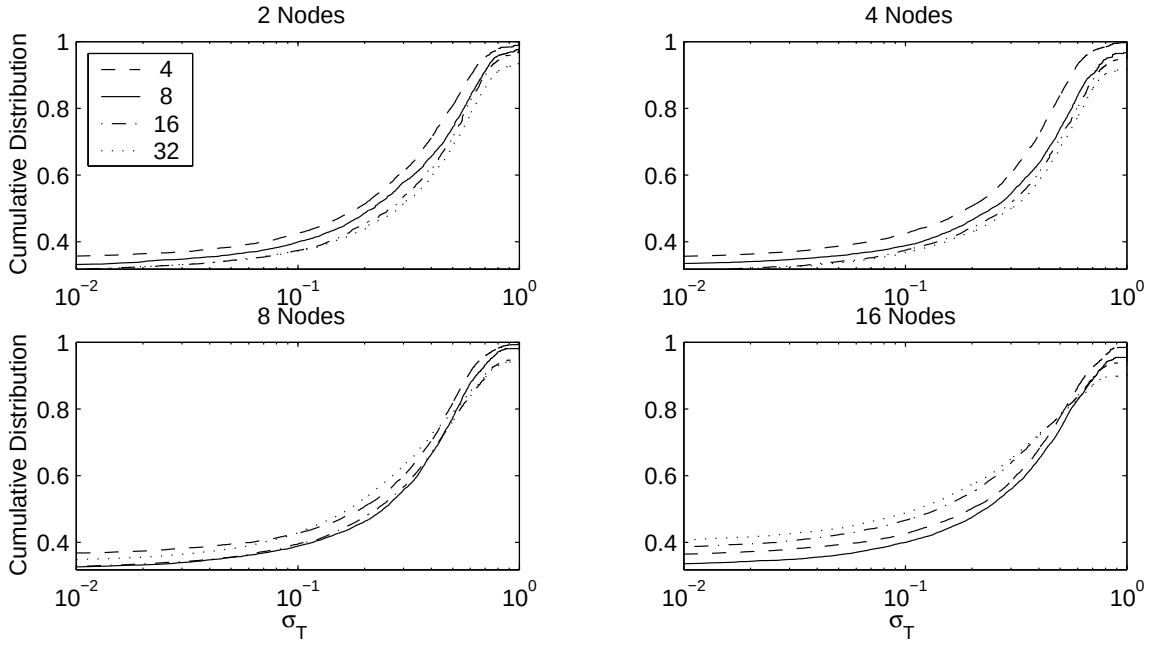

Figure 6: σ_T Distributions ($0.01 \le \sigma_T \le 1$) as number of nodes and epochs is varied, Diabetes 50/50 training/testing

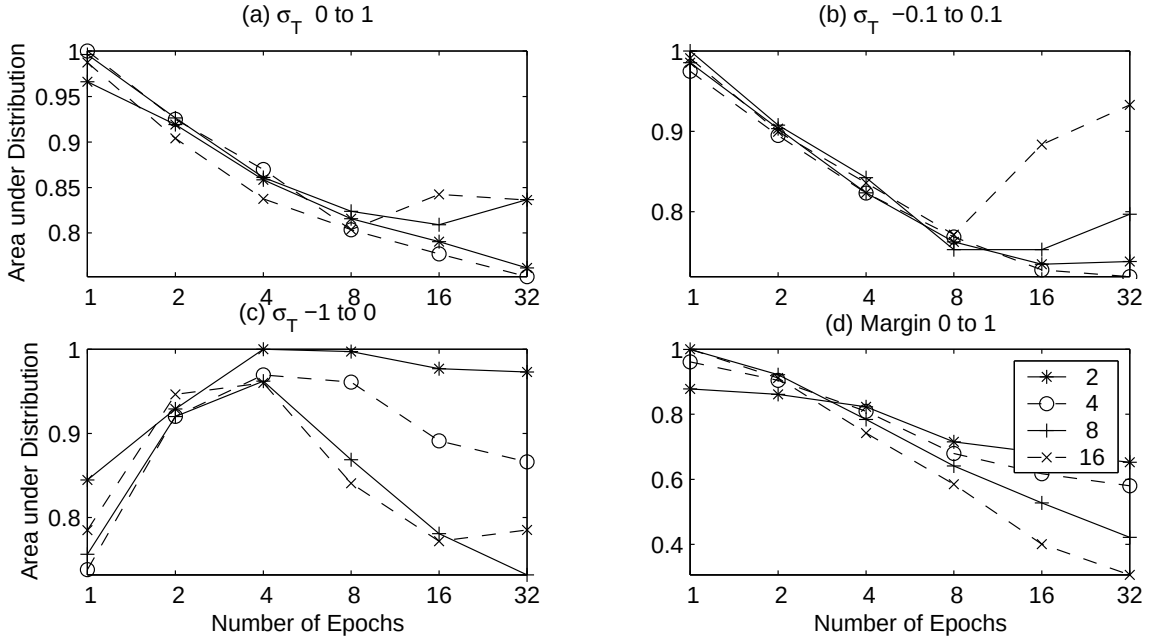

Figure 7: Area under various distributions, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

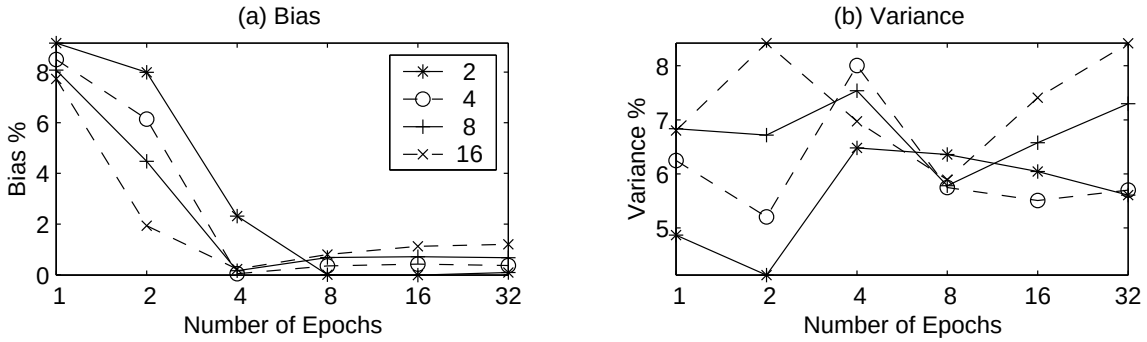

Figure 8: Bias and Variance, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

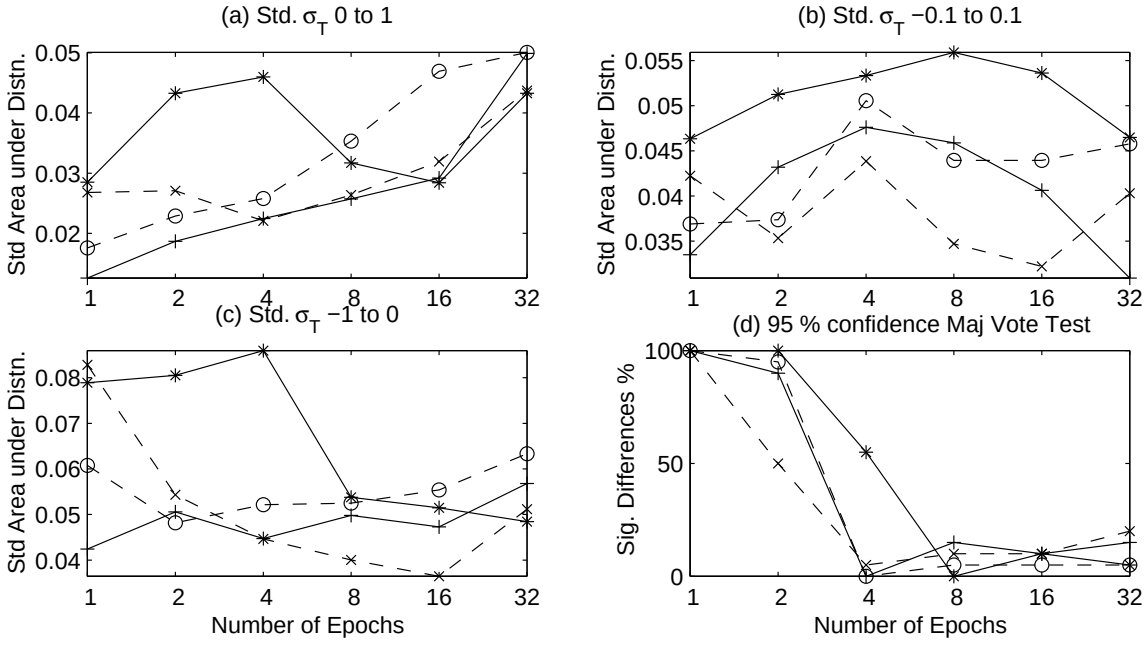

Figure 9: Standard deviations of area under σ_T distribution and percentage significant differences of Majority vote compared to best error rate, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

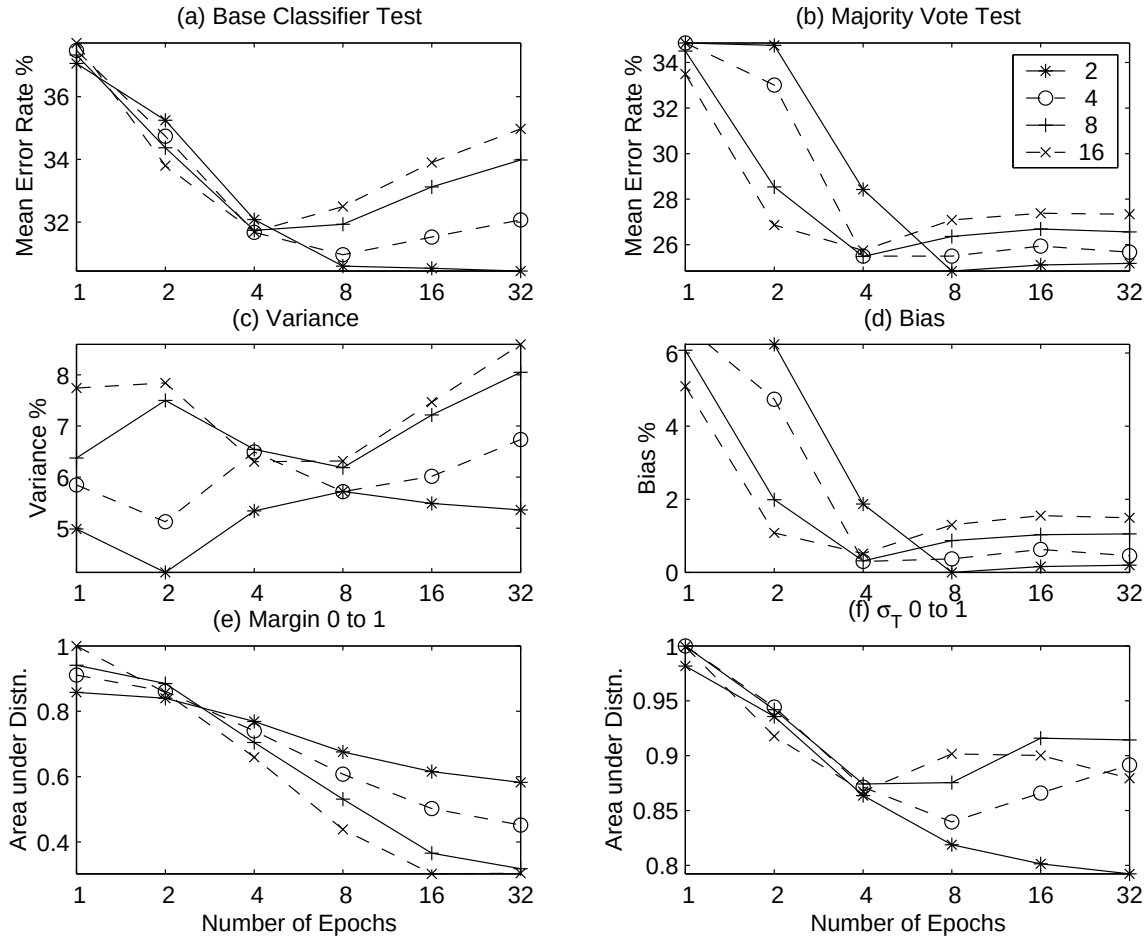

Figure 10: Diabetes 20/80 for 1-32 epochs, 2-16 nodes

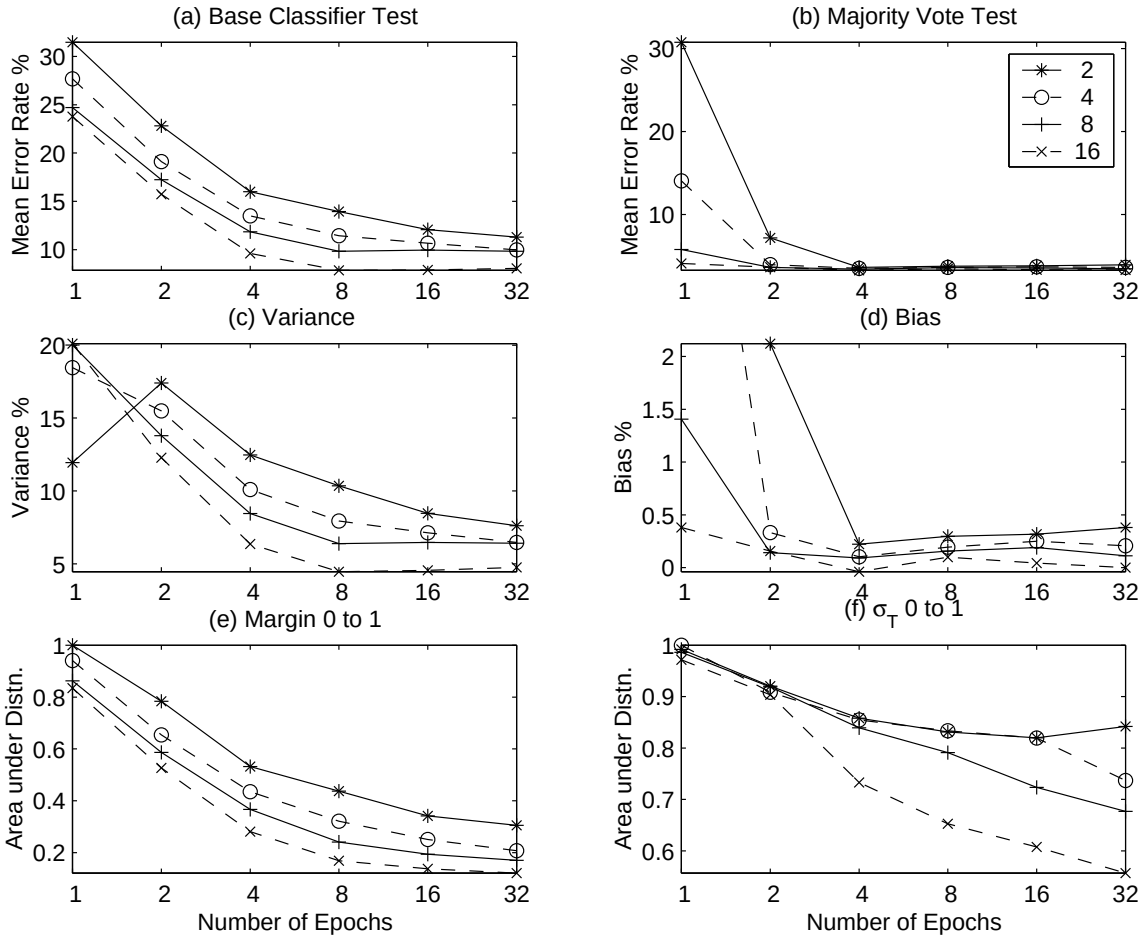

Figure 11: Cancer 50/50 for 1-32 epochs, 2-16 nodes

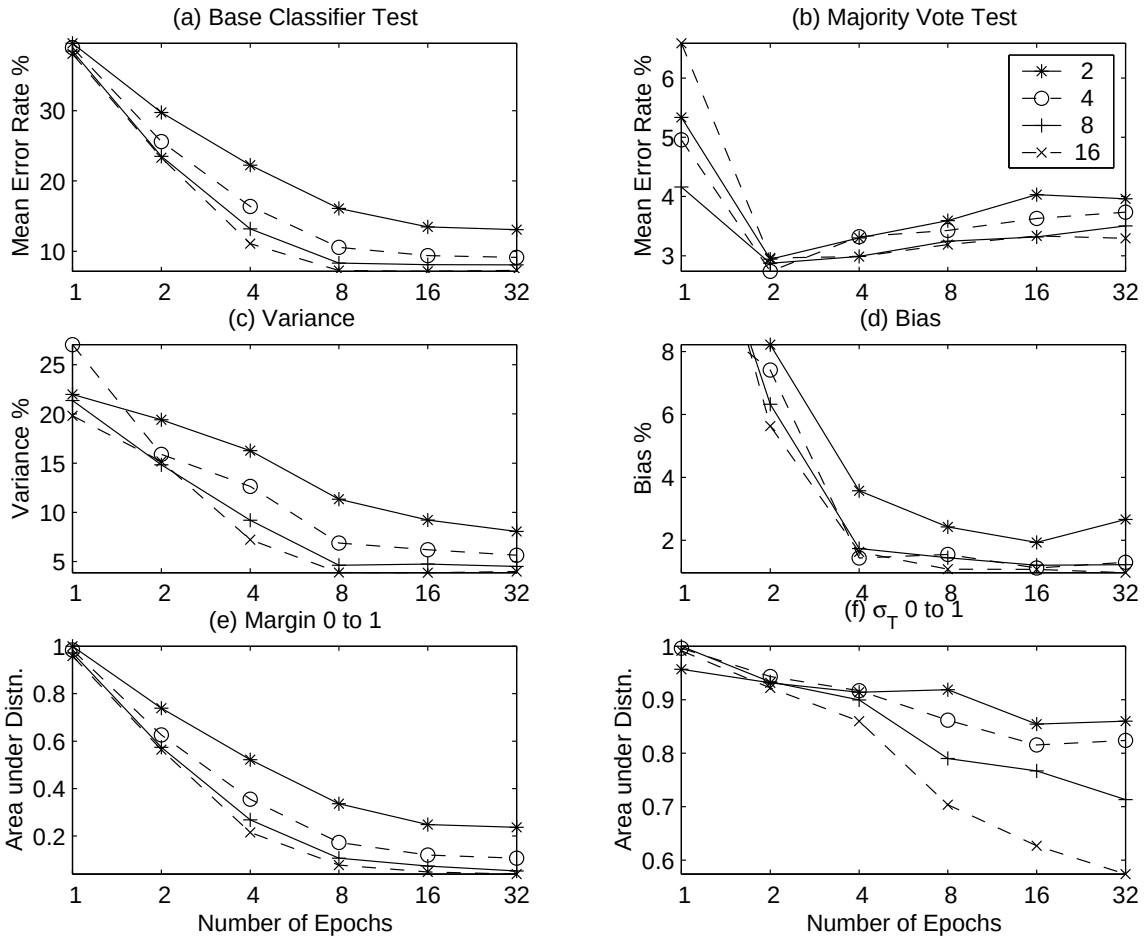

Figure 13: Twonorm 300 training patterns for 1-32 epochs, 2-16 nodes

1	2	3	class
1	1	1	1
-1	1	1	-1
1	-1	1	-1
-1	-1	1	1
1	1	-1	1
-1	1	-1	-1
1	-1	-1	-1
-1	-1	-1	-1

Table 1: Truth table for Example 1

Dataset	Maj. Vote $(-0.1 \le \sigma_T \le 0.1)$,	Base Classifier $(-0.1 \le \sigma_T \le 0.1)$,	Maj. Vote $(0 \le \sigma_T \le 1)$	Base Classifier $(0 \le \sigma_T \le 1)$
Diabetes	0.96	0.99	0.96	0.99
Cancer	0.95	0.86	0.86	0.93
Vote	0.44	0.18	0.60	0.85
Credita	0.30	0.21	0.78	0.97
Heart	0.43	0.44	0.84	0.97
Card	0.38	0.24	0.83	0.91
(Mean)	**0.58**	**0.49**	**0.81**	**0.94**

Table 2: Correlation coefficient between area under σ_T distribution $(-0.1 \le \sigma_T \le 0.1)$, $(0 \le \sigma_T \le 1)$ and test error rate as number of epochs is varied for 50/50 Training/Testing Natural Data

Dataset	Maj. Vote ($-0.1 \le \sigma_T \le 0.1$),	Base Classifier ($-0.1 \le \sigma_T \le 0.1$),	Maj. Vote ($0 \le \sigma_T \le 1$)	Base Classifier ($0 \le \sigma_T \le 1$)
Diabetes	0.94	0.97	0.97	0.99
Cancer	0.91	0.81	0.76	0.95
Ion	-0.08	-0.15	0.91	0.94
Heart	0.49	0.05	0.55	0.82
Card	0.17	-0.10	0.67	0.79
(Mean)	**0.49**	**0.32**	**0.77**	**0.90**

Table 3: Correlation coefficient between area under σ_T distribution ($-0.1 \le \sigma_T \le 0.1$), ($0 \le \sigma_T \le 1$) and test error rate as number of epochs is varied for 20/80 Training/Testing Natural Data

Dataset	Maj. Vote ($-0.1 \le \sigma_T \le 0.1$),	Base Classifier ($-0.1 \le \sigma_T \le 0.1$),	Maj. Vote ($0 \le \sigma_T \le 1$)	Base Classifier ($0 \le \sigma_T \le 1$)
Ringnorm	0.30	-0.11	0.81	0.92
Threenorm	0.41	0.45	0.85	0.86
Twonorm	0.93	0.61	0.21	0.89
(Mean)	**0.55**	**0.32**	**0.62**	**0.89**

Table 4: Correlation coefficient between area under σ_T distribution ($-0.1 \le \sigma_T \le 0.1$), ($0 \le \sigma_T \le 1$) and test error rate as number of epochs is varied for 300/3000 Training/Testing patterns Artificial Data

1 T. G. Diettrich, Ensemble Methods in Machine Learning, Proc. of 1st Int. Workshop on Multiple Classifier Systems, Cagliari, Italy, June, Lecture Notes in Comp. Science, Springer Verlag, 2000, 1-15.

2 L. Breiman, Bagging Predictors, Machine Learning, 24(2), (1997) 123-40.

3 Y. Freund, R.E. Schapire. A decision-theoretic generalisation of on-line learning and an application to boosting, J. of Computer and System Science, 55(1), (1997)119-139.

4 J. Friedman, T Hastie, and R Tibshirani, Additive Logistic Regression: A statistical view of Boosting, Annals of Statistics 28(2) (2000) 337-374.

5 E. M. Kleinberg, On the algorithmic implementation of stochastic discrimination, PAMI-22 (5), (2000) 473-490.

6 S. L. Hurst, *The Logical Processing of Digital Signals.* New York: Crane-Russak, 1978.

7 B. J. Falkowski, M.A.Perkowski, Effective Computer Methods for the Calculation of Rademacher-Walsh Spectrum for Completely and Incompletely Specified Boolean Functions. *IEEE Trans. on Computer-Aided Design* 11(10), (1992) 1207-1226.

8 S. L. Hurst, D. M. Miller, J. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, 1985.

9 A. N. Tikhonov, V. A. Arsenin, Solutions of ill-posed problems, Winston & Sons, Washington, 1977.

10 T. Windeatt, G. Ardeshir., Boosted Tree Ensembles for Solving Multiclass Problems, Proc. 3rd Int. Workshop Multiple Classifier Systems, Cagliari, Italy, Lecture notes in computer science, Springer-Verlag, 2002 pp 42-51.

11 J. Rissanen, Modeling by shortest data description, Automatica, 14, (1978) 465-471.

12 V. Koltchinskii, Rademacher penalties and Structural Risk Minimisation, IEEE Trans. On Information Theory 47(5), (2001) 1902-1914.

13 V. Vapnik, Statistical Learning Theory, Wiley, 1998.

14 T. Windeatt, R. Ghaderi., Binary labelling and Decision Level Fusion, Information Fusion 2(2), (2001) 103-112.

15 T. G. Dietterich, G. Bakiri, Solving Multi-class learning problems via ECOC, J. of Artificial Intelligence Research, 2, (1995) 263-286.

16 T. Windeatt, R. Ghaderi, Coding and Decoding Strategies for multiclass learning problems, Information Fusion, 2003, to appear.

17 J. C. Muzio, S. L. Hurst, (1978). The computation of complete and reduced sets of orthogonal spectral coefficients for logic design and pattern recognition purposes, *Comput. Electric. Engrg.* 5, (1978) 231-249.

18 T. Windeatt, R. Tebbs, Spectral technique for hidden layer neural network training, Pattern Recognition Letters, Vol.18(8) (1997) 723-731.

19 T. G. Dietterich, Approx. statistical tests for comparing supervised classification learning algorithms, Neural Computation 10, (1998) 1895-1923.

20 R. E. Schapire, Y. Freund, P. Bartlett, Boosting the Margin: a new explaination for the effectiveness of voting methods, The Annals of Statistics 26(5), (1998) 1651-1686.

21 V. Koltchinskii, D. Panchenko, Empirical margin distributions and bounding the generalisation error of combined classifiers, The Annals of Statistics 30(1), (2002) 1-50.

22 G. Ratsch, T. Onoda, K R Muller, Soft Margins for Adaboost, Machine Learning 42(3), (2001) 287-320.

23 G. James, Variance and Bias for General Loss Functions, Machine Learning, 2003, to appear.

24 E. B. Kong, T. G. Dietterich, Error- Correcting Output Coding corrects Bias and Variance, 12th Int. Conf. Machine Learning, San Francisco, (1995) 313-321.

25 L. Breiman, Arcing Classifiers, The Annals of Statistics 26(3), (1998) 801-849.

26 M. Skurichina, L. I. Kumcheva, R. P. W. Duin, Bagging and Boosting for the Nearest Mean Classifier: Effects of Sample Size on Diversity and Accuracy., Proc. 3rd Int. Workshop Multiple Classifier Systems, Cagliari, Italy, Lecture notes in computer science, Springer-Verlag, 2002 pp 62-71.

27 L. Prechelt, Proben1: A set of neural network Benchmark Problems and Benchmarking Rules, Tech Report 21/94, Univ. Karlsruhe, Germany, 1994.

28 C.J. Merz , P. M. Murphy, UCI repository of machine learning databases, 1998

29 T. Windeatt, Recursive Partitioning for combining multiple classifiers, Neural Processing Letters 13(3), (2001) 221-236.

30 T. Windeatt, R. Ghaderi, Output Coding, in Pattern Recognition and String Matching, ed. D. Chen X. Cheng, Kluwer, 2003, to appear.

31 J. M. Galey, R. E. Norby, J. P. Roth, Techniques for the Diagnosis of Switching Circuit Failures. *IEEE Trans. Comms. and Electronics* 83(74), (1964) 509-514.

32 S. B. Akers, B. Krishnamurthy. Test Counting: A Tool for VLSI Testing. *IEEE Design and Test of Computers*, October 1989, (1989) 58-77.

33 H. Fujiwara, *Logic Testing and Design for Testability*. MIT Press, 1985.

34 E. J. McCluskey, Minimisation of boolean functions, Bell Syst. Tech. J., Vol 35(5) (1956) 1417-1444.

35 S. Muroga. *Threshold Logic & its Applications,* Wiley, 1971.

C.V.

Terry Windeatt received the BSc degree in Applied Science from University of Sussex, followed by M.Sc. in Electronic Engineering from University of California, B.A.(CNAA) in theology and PhD degree from University of Surrey, U.K. After lecturing in Control Engineering at Kingston University, UK, he went to live and work in the USA for eight years. He worked on Intelligent Systems in the Research and Development Departments of General Motors and Xerox Corporation in Rochester, NY. His industrial R&D experience is in modelling/simulation for intelligent automotive and office-copying applications. He returned from the United States in 1984 to join the Department of Electrical and Electronic Engineering at the University of Surrey, where he now lectures in Machine Intelligence. He has worked on various research projects in the Centre for Vision, Speech and Signal Processing, and his current research interests include Pattern Recognition, Neural Nets and Computer Vision.

Figure Captions

Figure 1: Matrix multiplication to calculate spectral coefficients in Example 1

Figure 2: Class 1 patterns after applying equation (3) in Example 1

Figure 3: Karnaugh Map representation for Example A2 showing σ as double-headed arrows

Figure 4: Base classifier and Majority Vote training and testing error rates, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

Figure 5: Margin distributions ($0 \le$ Margin ≤ 1), Diabetes 50/50 for 4-32 epochs, 2-16 nodes

Figure 6: σ_T Distributions ($0.01 \le \sigma_T \le 1$) as number of nodes and epochs is varied, Diabetes 50/50 training/testing

Figure 7 Area under various distributions, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

Figure 8: Bias and Variance, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

Figure 9 Standard deviations of area under σ_T distribution and percentage significant differences of Majority vote compared to best error rate, Diabetes 50/50 for 1-32 epochs, 2-16 nodes

Figure 10: Diabetes 20/80 for 1-32 epochs, 2-16 nodes

Figure 11 Cancer 50/50 for 1-32 epochs, 2-16 nodes

Figure 12: Twonorm 300 training patterns for 1-32 epochs, 2-16 nodes