

Designing Classifier Ensembles with Constrained Performance Requirements

Weizhong Yan⁺ and Kai F. Goebel^{*}

ABSTRACT

Classification requirements for real-world classification problems are often constrained by a given true positive or false positive rate to ensure that the classification error for the most important class is within a desired limit. For a sufficiently high true positive rate, this may result in the set-point being located somewhere in the flat portion of the ROC curve where the associated false positive rate is high. Any further classifier design will then attempt to reduce the false positive rate while maintaining the desired true positive rate. We call this type of performance requirements for classifier design the constrained performance requirement. This type of performance requirements is different from the accuracy maximization requirement and thus requires different strategies for classifier design. This paper is concerned with designing classifier ensembles under such constrained performance requirements. Classifier ensembles are one of the most significant advances in pattern recognition/classification in recent years and have been actively studied by many researchers. However, not much attention has been given to designing ensembles to satisfy constrained performance requirements. This paper attempts to identify and address some of design related issues associated with the constrained performance requirement. Specifically, we present a design strategy for designing neural network ensembles to satisfy constrained performance requirements, which is illustrated by designing a real-world classification problem. The results are compared to those from conventional design method.

Keywords: classification; classifier ensembles; neural networks; constrained performance; diversity; accuracy; ROC;

1. INTRODUCTION

Classifier design is the task of developing a classification system with performance that satisfies the specified performance requirements. Two primary factors determine how difficult a classifier design problem is. They are: 1) the complexity of the problem itself and 2) the required classification performance for the problem. These two factors are interrelated (not independent). Any classification problem, no matter how difficult it is, can be simple if the required classification performance is low. On the other hand, if the required classification performance is significantly high, an otherwise simple classification problem can become a difficult one.

Many factors contribute to the complexity of a classification problem. In particular, design data with low class separability in the feature space may be the main attribute of complexity. In addition, high dimension of feature space and limited size of available training data may further complicate the classification problems.

Performance requirements posed to a classification problem vary from application to application. While classification accuracy is the most widely used performance measure in specifying performance requirements, in the field of computer aided diagnosis (CAD) and as well mechanical diagnostics (MD), where misclassifying one class may have much more

⁺ yan@research.ge.com; phone: (518) 387-5704; fax: (518) 387-6104; <http://www.rpi.edu/~yanw/>; GE Global Research Center, K1-5B34B, One Research Circle, Niskayuna, NY 12309, USA

^{*} goebelk@research.ge.com; phone: (518) 387-4194; fax: 518 387-6104; <http://best.me.berkeley.edu/~goebel/kai.html>; GE Global Research Center, K1-5C4A, One Research Circle, Niskayuna, NY 12309, USA

severe cost consequences than others, classifier performance is typically specified by minimizing one type of error while maintaining another type of error to a pre-determined level. We call this type of performance requirements for classifier design the *constrained performance requirements*. Misclassification cost is another performance measure typically used in this situation ¹. However, misclassification cost requires knowledge of explicitly defined cost for each of the classes, which often time is not available and is difficult to estimate in many applications.

Traditionally, design of classification systems is to empirically choose a single classifier through experimental evaluation of a number of different ones. The parameters of the selected classifiers are then optimized so that the specified performance is met. It has been well recognized that single classifier systems have limited performance. Thus for certain real-world classification problems, this single classifier design approach may fail to meet the desired performance even after all parameters/architectures of the classifier have been fully optimized. In these cases, using classifier ensembles, one of the most significant advances in pattern classification in recent years, proves to be more effective and efficient ². An ensemble of classifiers is a set of individually trained classifiers whose individual decisions are combined in some way ³. By taking advantage of complementary information provided by the constituent classifiers, classifier ensembles offer improved performance, i.e., they are more accurate than the best individual classifier.

Current practice in ensemble design focuses on generating modestly accurate but diverse individual classifiers. Numerous empirical studies have demonstrated that such a design strategy ensures an improved performance when the performance is measured as classification accuracy ⁴. However, not much attention has been devoted to classification problems with constrained performance requirements. We argue that constrained performance requirements are distinct from the conventional accuracy maximization requirement and thus require different design strategies. Studies on identifying and addressing the design related issues associated with constrained performance requirements are necessary. Specifically, we want to know whether generating accurate and diverse classifiers for ensembles, which proves to work well for accuracy improvement, guarantees that an ensemble performs better with respect to the constrained performance requirements. More importantly, we need strategies on how to design an ensemble for such classification problems. This paper is an attempt towards addressing some of these issues.

Specifically, through designing a real-world classification problem for illustration, we demonstrate that neural network ensembles can be effective in improving classification performance for problems with low class separability. More importantly, we present a design strategy for designing neural network ensembles to satisfy constrained performance requirements. The rest of this paper is organized as follows. Section 2 provides an overview of classifier ensembles in general and neural network ensembles in particular. Section 3 formulates the constrained performance requirements. Details of the proposed design strategies are discussed in Section 4. Section 5 shows the experimental results where a real-world classification problem is designed using the proposed approach. The results are compared with those from using conventional approach. Section 6 concludes the paper with a few final thoughts.

2. CLASSIFIER ENSEMBLES

Classifier ensembles have been considered as one of the most active research directions in machine learning ³. An ensemble of classifiers is a set of classifiers whose decisions or outputs are combined to arrive at a final classification decision. The individual classifiers, also called component classifiers, constituent classifiers, and base classifiers, in the ensemble can be any type of classifiers, preferably “unstable” classifiers like neural networks and decision trees. “Unstable” here means the learning algorithms that a small change in the training samples will lead to a classifier with different classification performance ⁵. Neural network ensembles where neural network classifiers are used as component classifiers are one of the ensembles that have been actively studied in recent years ^{6,7,8}.

Many studies (both theoretical and empirical) have proved that classifier ensembles are more accurate than the individual classifiers in the ensemble ^{9,10}. As a result, classifier ensembles have been used for various application domains ranging from optical character recognition (OCR) ¹¹ to seismic signal classification ¹². One of the recent findings is that in order for ensembles to be more accurate than the constituent classifiers in the ensemble, individual classifiers need to be modestly accurate, but, most importantly, need to be diverse (making classification errors on different examples) ⁹. Therefore, one of the most active areas of research in ensembles has been centered on how to generating accurate and diverse individual classifiers for ensembles. Obviously, combining methods also play an important role in improving

performance, however, studies ^{5, 10} have shown that a simple average of the outputs of individual classifiers is an effective combining scheme.

There are many methods to generate diverse individual classifiers for ensembles. These methods can be broadly categorized into three groups as follows.

- 1) Using different structure or architecture for individual classifiers. For neural network ensembles, the individual networks can have different number of hidden layers and different number of hidden neurons, and/or are trained with different initial weights and biases. For decision tree ensembles, different number of nodes may be used.
- 2) Using different training samples for individual classifiers. The two most popular methods in this group are bagging and boosting. In bagging (short for “bootstrap aggregating”) ¹³, training set for each classifier is a bootstrap replicate of the original training set, which is obtained by uniformly sampling, with replacement, the original training data. In boosting ^{14, 15}, individual classifiers are trained sequentially (in a series). The training set for a k^{th} classifier in the series is chosen based on the performance of classifiers 1 through $k-1$. The probability of selecting an example for k^{th} classifier depends on how often that example was misclassified by all $k-1$ classifiers before it.
- 3) Using different subsets of features for individual classifiers. Each individual classifier is trained with a different subset of features. The feature subsets can be obtained by random selection ¹⁶, or input decimation ¹⁷, or using genetic algorithms (GA) ^{18, 19}.

3. CONSTRAINED PERFORMANCE REQUIREMENTS

Overall classification accuracy (or alternatively overall classification error) is the most popular measure of classifier performance. Accuracy has been almost exclusively used in design and evaluation of classifier ensembles. The underlying assumption of the accuracy as a classifier performance measure is that the classification errors for all classes have equal cost consequences, which rarely holds for real-world applications. In real applications, there are a great number of problems, especially in CAD and MD fields, which have different misclassification costs for different classes. However, exact costs associated with different misclassification are rarely known and are difficult to estimate. As a result, these types of problems typically have the performance requirements being specified in the form of specific level of error for one or both types of errors which we called “constrained performance requirements” in this paper. In decision-making, this type of design requirements is called the *Neyman-Pearson decision criterion* ²⁰.

Before formulating the constrained performance requirements, we first define the two types of classification errors (false positive and false negative) and ROC curves that show tradeoffs between the two types of errors for a classifier. We restrict our discussion to two-class classification problems where the two classes are referred to as the positive and negative classes.

		Classes assigned by Classifier	
		Negative	Positive
True Classes	Negative	N⁰⁰	N⁰¹
	Positive	N¹⁰	N¹¹

Figure 1 - A typical confusion matrix for 2-class classification problems

Consider a classification problem specified by a test set, $\{\bar{x}_i, y_i\}$, $i = 1, 2, \dots, l$, where $\bar{x}_i \in R^D$ is the i^{th} input vector and $y_i \in \{0, 1\}$ (“0” stands for negative class and “1” for positive class) is the corresponding class label. Outputs of the classifier to the test data set are typically summarized in a *confusion matrix* (also called *contingency table*) as shown in Figure 1.

A classifier can make two types of errors, namely, false positive error and false negative error. In the field of statistical hypothesis testing, they are referred as Type I and Type II errors. $P(\text{false positive}) = P(\text{classified positive} | \text{negative})$, and $P(\text{false negative}) = P(\text{classified negative} | \text{positive})$. From the confusion matrix defined above, the error rates, false negative rate (FNR) and false positive rate (FPR), can be calculated as follows.

$$FNR = \frac{N^{10}}{N^{10} + N^{11}} \quad (1.a)$$

$$FPR = \frac{N^{01}}{N^{00} + N^{01}} \quad (1.b)$$

It is worthwhile to note that, for a given classifier, the two types of errors defined above vary with decision actions. In fact, by varying the decision threshold, one can reduce one type of errors to any arbitrarily small number through sacrificing another type of errors.

The tradeoff between the two types of errors for a classifier can be better depicted by ROC curves. ROC (short for Receiver Operating Characteristic or Relative Operating Characteristic) curves represent the behavior of a classifier without regard to class distribution or error cost. Hence ROC curves are a true representative of classifier performance. In ROC space, the true positive rate, TPR (TPR=1-FNR), is plotted on the Y-axis and the false positive rate, FPR, is plotted on the X-axis, where the TPR is commonly referred to as “sensitivity” while (1-FPR) is called “specificity”. A point in ROC space corresponds to a (FPR, TPR) pair of a classifier. Typical ROC curves are shown in Figure 2 where the three ROC curves represent three different classifiers. Classifiers with ROC curves located in the upper-left corner in ROC space are better because they represent classifiers that have lower false positive rate and higher true positive rate than the classifiers below them¹. ROC curves are typically generated by varying either some parameters of the classifier (e.g. threshold in neural networks) or costs (e.g. in decision trees).

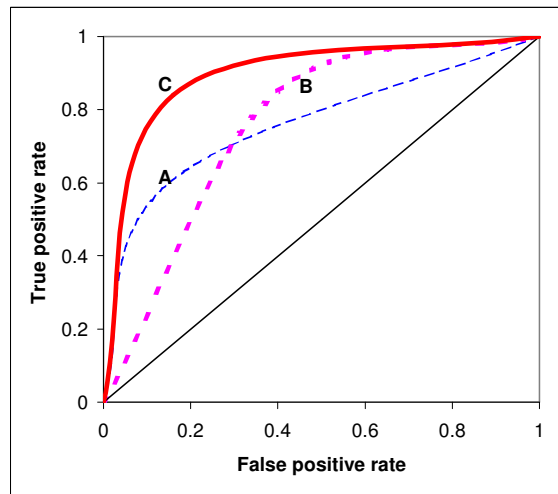


Figure 2 - Typical ROC curves

Constrained performance requirements can be specified in different forms, depending on which rate (FPR or FNR or both) needs to be set to a predetermined level. For presentation convenience, for the rest of this paper, we only consider

the constrained performance requirement as being to minimize FPR while maintaining TPR (= 1-FNR) to a predetermined large number. Let the predetermined large number be α , the constrained performance can then be expressed as

$$\min(FPR) \text{ subject to } TPR \geq \alpha \tag{2}$$

Alternatively, the constrained performance requirement is to minimize the quantity [Therrien (1989)]

$$\chi = FPR + \lambda \cdot (TPR - \alpha) \tag{3}$$

where λ is a Lagrange multiplier.

Mathematically, the constrained performance requirement is a constrained optimization problem that is somewhat difficult to solve. As a result, a practical approach to design a classifier to satisfy a constrained performance requirement is to generate a ROC curve first for the classifier by varying either the decision threshold, or some classifier structure parameters. Then find the corresponding FPR from the ROC curve based on the predetermined TPR number.

The constrained performance requirement (Equation 2) can be graphically represented in ROC space as shown in Figure 3. With this requirement, the designer will attempt to obtain a better classifier by moving the interception point of the ROC curve and the line defined by $TPR = \alpha$ towards the Y-axis along line $TPR = \alpha$ (i.e., from point A to point B in Figure 3). This design strategy is different from that when accuracy maximization is the performance requirement. In the latter case, the designer is trying to obtain a better classifier by moving the ROC curve as close to point [0,1] as possible in the ROC space. As can be seen in Figure 3, if the predetermined TPR, α , is significantly high (close to 1.0), the interception will occur in the “flat” portion of the ROC curve, which implies that a great amount of accuracy increase is needed to achieve even a small amount of reduction of FPR.

In summary, the constrained performance requirement has its own characteristics and is distinct from the accuracy maximization requirement, which deserves different design strategies in classifier design.

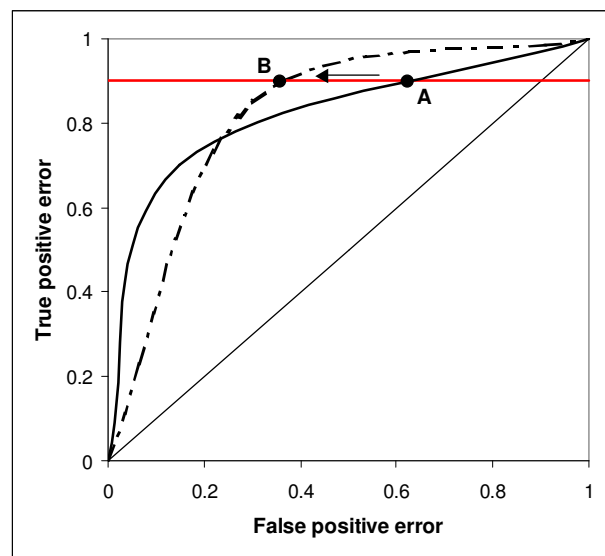


Figure 3: Graphic illustration of constrained performance in ROC space

4. DESIGNING NN ENSEMBLES FOR CONSTRAINED PERFORMANCE REQUIREMENTS

At this time, a systematic method for designing classifier ensembles is still an open topic ²¹. Neural network ensembles, and other classifier ensembles alike, are typically designed heuristically in two steps: first generating individual classifiers, and then combining the outputs of the individual classifiers, for example, by simply averaging. In the classifier generation step, the designer focuses on generating individual classifiers that are modestly accurate but as diverse as possible. The underlying hypothesis is that as long as the individual classifiers are reasonably accurate and diverse, the resulting ensemble will be more accurate. Although general relationship between ensemble accuracy and the diversity is still not clear, numerous studies have proved that such design approach (generating accurate and diverse individual classifiers) can often time yield an ensemble that is more accurate than individual classifiers ⁶. For classification problems with constrained performance requirements, however, our goal is to minimize one type of error while maintaining another error to a pre-defined level. Simply generating diverse individual classifiers may not result in an ensemble that optimally meets the performance requirements. Instead, we argue that the individual classifiers need to be designed by directly targeting to meet the constrained performance requirements in addition to having the normal properties, i.e., reasonable accuracy and high possible diversity. To that end, we embed constrained performance requirements into the feature selection process to ensure that individual classifiers have performance bias towards the requirement.

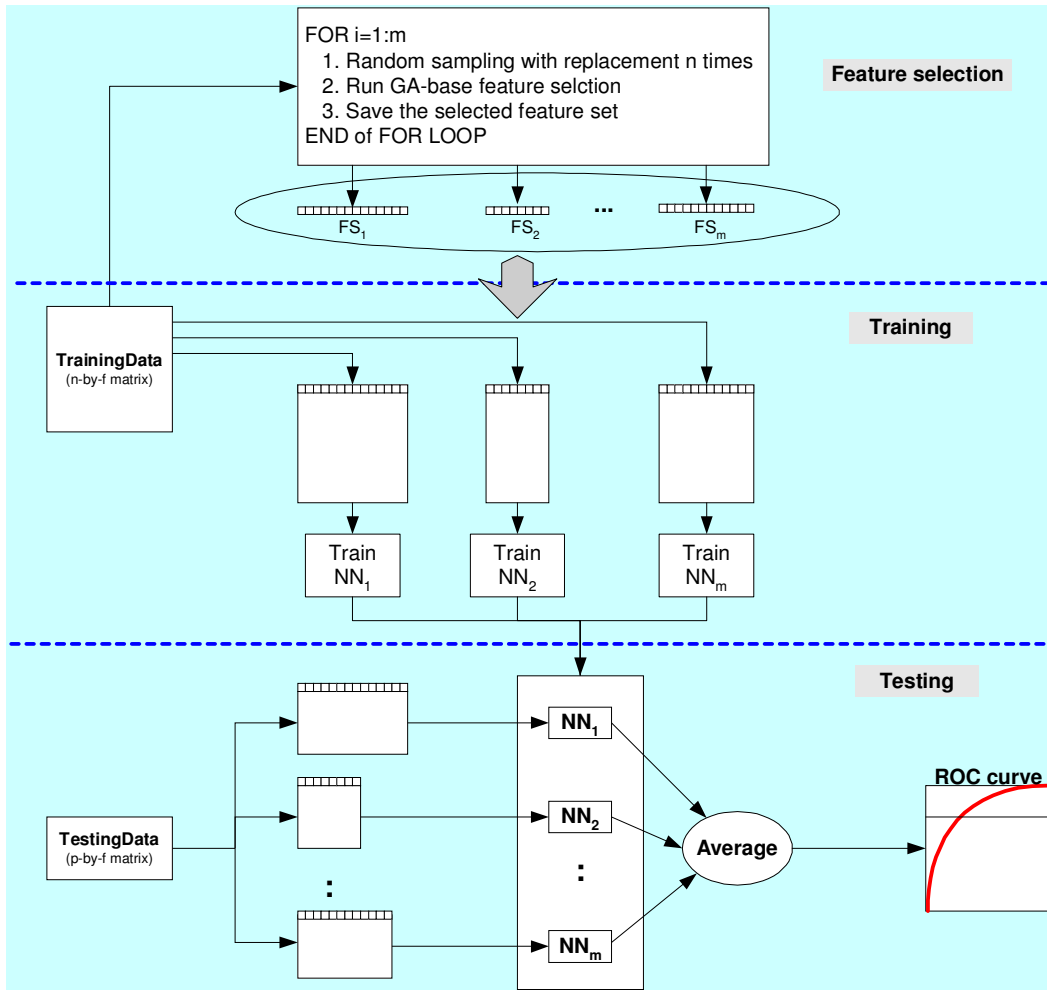


Figure 4: Overall structure of the proposed method

We take the following three diversity-promoting measures in designing individual neural networks:

- 1) Each neural network in the ensemble has a different number of hidden neurons.
- 2) Each neural network in the ensemble is trained with a subset of data that is randomly sampled from the original training data.
- 3) Each neural network in the ensemble uses a different subset of features, which is selected using one run of genetic algorithms (GA) based feature selection.

Unlike conventional GA-based feature selection for ensembles^{18,19}, where the fitness function is either accuracy or some combination of accuracy and diversity of the classifiers, we set the fitness function of GA to be exactly the constrained performance requirement, that is, to $\min(FPR)$ subject to $TPR \geq \alpha$ during feature selection process.

The idea is that such performance-seeking features will help the individual networks that are trained based on the selected features to perform bias towards the constrained performance requirement. Hence, the ensembles of these individual networks will be better in meeting the constrained performance requirement. The overall architecture of our approach in designing neural network ensembles with constrained performance requirements is illustrated in Figure 4, where m individual neural networks are shown in the ensemble. The design consists of three main steps: 1) GA-based feature selection, 2) network training, and 3) classifier testing. In the feature selection step, we run GA-based feature selection m times to obtain m sets of features, each of which is used for one individual neural network. Each GA run uses a different data set in addition to using a different random initial population. The data sets are obtained by randomly sampling, with replacement, the original data set n times, where n is the number of examples in the training set. This data-generating scheme for each GA run is similar to “bagging”. In the training step, each neural network with different number of hidden neurons is trained independently using the entire training data. The number of inputs for each individual neural network is the number of features that GA selected. The trained networks are then tested using the testing data set. The outputs of the networks for each case are averaged to arrive at the output of the ensemble corresponding to the case. By varying decision threshold and applying against the ensemble outputs, a set of TPR-FPR pairs is obtained, thus the ROC curve can be generated.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The real-world application concerned in this paper is an automated data analysis system for non-destructive inspection. To perform classification, 370 features from different domains were extracted. For classifier design, 5600 examples representing different conditions of the target object are used. Of the 5600 examples, 2600 are for defected condition while 3000 are for normal condition. The classification performance requirement is less than 50% false positive rate with the true positive rate of greater than or equal to 98%.

During our preliminary design, we tried several single classifier systems including neural networks, support vector machine, and decision tree for the classification problem concerned. The false positive rates for those single classifier systems are around 75% with corresponding true positive rate of 98%, which does not meet the design requirement.

The neural network ensemble for this application is designed in two methods, namely, the conventional accuracy maximization method and the proposed constrained performance maximization method. In both methods, ten (10) neural networks are used as the base classifiers. By using 10 networks we followed the suggestion of Opitz and Maclin (1999)²² who reported that ensembles with as few as 10 base classifiers are adequate to sufficiently reduce error. The networks are fully connected feedforward type with a single hidden layer. To increase the diversity of those individual classifiers, each network uses different number of hidden neurons. Additionally, each network uses different training data that are obtained by randomly sampling, with replacement, from the original data set. Furthermore, each network uses different features that are selected through a GA-based feature selection process.

The main difference between the two methods is the use of the different fitness functions for GA feature selection. In the conventional design, the fitness function of GA is the training set accuracy. The proposed design, on the other hand, uses the very performance requirement, i.e., the false positive rate corresponding to true positive rate of 98%, as the fitness function for GA feature selection.

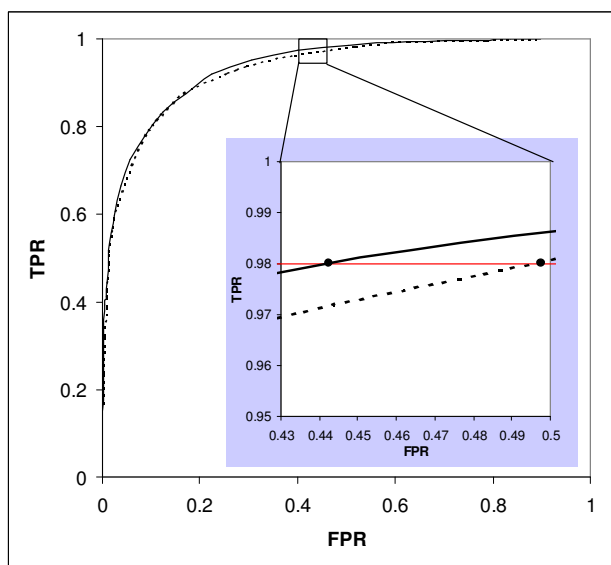


Figure 5: Comparison of ROC curves from 2 designs

The ROC curves of the two design methods are shown in Figure 5 where the dashed curve is for the conventional design method while the solid curve is for the proposed method. For better appreciating the difference between the two ROC curves at TPR level of 98%, a small window of the curves is magnified as shown in the shaded box of Figure 5. At TPR level of 98%, the false positive rates for conventional and proposed design methods are 49.7% and 44.2%, respectively. In other words, the proposed method yields approximately 6% less false positive rate than the conventional method does.

Table 1: Comparison of individual accuracy and ensemble performance

Design methods	Individual classifier accuracies				Ensemble performance		
	min	max	mean	std	accuracy	diversity	<i>FPR @ TPR of 98%</i>
Conventional (accuracy maximization)	0.8156	0.8549	0.8348	0.0137	0.8555	0.4281	0.497
Proposed (constrained performance maximization)	0.8077	0.8509	0.8290	0.0166	0.8509	0.5735	0.442

To further compare the two design methods and to identify what contributes to the difference, we calculate the accuracies and diversities of the 10 individual networks. The accuracy for each individual classifier is determined based on individual ROC curves. The diversity of individual classifiers is calculated based on ambiguity defined by ¹⁰. That is, the ambiguity of i^{th} classifier on k^{th} example is $a_i(k) = [V_i(k) - \bar{V}(k)]^2$, where $V_i(k)$ and $\bar{V}(k)$ are the outputs of i^{th} classifier and the ensemble, respectively, on k^{th} example. Instead of showing accuracies for all 10 networks, we calculate and show in Table 1 the min, the max, the mean, and the standard deviation of accuracies of the 10 networks from the two designs. The calculated accuracies and diversity of each of the design methods are summarized in Table 1. For completion, the false positive rates obtained from ROC curves for both design methods are also shown in the last column of Table 1.

From Table 1 we can see that, even though the two design methods use the identical network architecture for individual classifiers and use the same strategies (feature selection) for promoting diversity, the two design methods yield different

accuracy and diversity. That is, while the conventional method yields more accurate classifiers (both individual and ensemble), the proposed method creates individual classifiers with higher diversity. The difference comes solely from the different fitness functions used in GA-based feature selection of the two design methods. It seems that the proposed method sacrifices accuracy for high diversity.

Also from Table 1, we can see that the proposed method is better in meeting the constrained performance goal since it shows lower false positive rate at TPR level of 98% comparing to the conventional design. Had accuracy maximization been our performance requirement, we would have taken the classifier designed using conventional method as the better design. We suggest this supports our claim that different design strategies/methods are required to better satisfy different performance requirements.

6. CONCLUSIONS

In this paper, we attempt to address issues associated with designing NN ensembles to satisfy constrained performance requirements. We point out that constrained performance requirements are distinct from the conventional accuracy maximization requirement and that the common design strategy for neural network ensembles, which focuses on generating modestly accurate but diverse individual classifiers, may not be sufficient for designing an classifier ensemble to optimally satisfy the constrained performance requirements. We proposed a new design strategy for designing neural network ensembles with constrained performance requirements. We also illustrate the new design strategy by designing a complex, real-world classification problem.

The experimental results of the real-world problem seem to indicate that the proposed design strategy can be a promising one for designing NN ensembles with constrained performance requirements. Much more work is needed before we can fully understand and address the issues discussed in this paper.

Specifically, future work should include investigating the feasibility of using cost-sensitive learning to design individual classifiers so that the resulting ensembles will better meet the constrained performance requirement. Ultimately, a systematic approach for designing ensembles to satisfy constrained performance requirements is needed.

REFERENCES

1. Yan, W.Z, Goebel, K.F., and Li, J.C. (2002), "Classifier performance measures in multi-fault diagnosis for aircraft engines", in Component and Systems Diagnostics, Prognostics, and Health Management II, Peter, K. Willett, & Thiagalingam Kirubarajan, Editors, Proceedings of SPIE Vol. 4733, pp88-97
2. Dietterich, T.G. (2000), "An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization", Machine learning, 40(2), pp139-158
3. Dietterich, T.G. (1999), "Machine learning research: four current directions", AI Magazine, Vol. 18, No. 4, pp97-136
4. Opitz, D.W and Shavlik, J. (1996), "Actively searching for an effective neural network ensemble", Connection Science, 8(3/4), pp337-353
5. Breiman, L. (1996a), "Stacked regression", Machine Learning, Vol. 24, No. 1, pp49-64
6. Opitz, D.W. and Maclin, R.F. (1997), "An empirical evaluation of bagging and boosting for artificial neural networks", Proceedings of International Conference on Neural Networks, Huston, Texas, Vol. 3, pp1401-1405
7. Liu, Y. and Yao, X. (1999), "Simultaneous training of negatively correlated neural networks in an ensemble", IEEE Transactions of Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 29, No. 6, pp716-725

8. Brown, G. and Wyatt, J. (2003), "The use of the ambiguity decomposition in neural network ensemble learning methods", Proceedings of the 12th International Conference on Machine Learning (ICML-2003), Washington, DC
9. Hansen, L. and Salamon, P. (1990), "Neural network ensembles", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 10, pp993-1001
10. Krogh, A. and Vedelsby, J. (1995), "Neural network ensembles, cross validation, and active learning", In Tesauro, G., Touretzky, D. & Leen, T. (Eds), Advances in Neural Information Processing Systems, Vol. 7, pp231-238, Cambridge, MA, MIT Press.
11. Mao, J. (1998), "A case study on bagging, boosting, and basic ensembles of neural networks for OCR", Proceedings of IEEE World Congress on Computational Intelligence, Vol. 3, pp1828-33
12. Shimshoni, Y. and Intrator, N. (1998), "Classification of seismic signals by integrating ensembles of neural networks", IEEE Transactions of Signal Processing, Vol. 46, No. 5, pp1194-1201
13. Breiman, L. (1996b), "Bagging predictors", Machine Learning, Vol.24, No.2, pp123-40
14. Schapire, R.E. (1990), "The strength of weak learnability", Machine Learning, Vol.5, No.2, pp197-227
15. Freund, Y. and Schapire, R.E. (1996), "Experiments with a new boosting algorithm", In Saitta, L. (Ed.), Proceedings of the 13th International Conference on Machine Learning, pp148-156
16. Ho, T.K. (1998), "The random subspace method for constructing decision forests", IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), pp832-844
17. Tumer, K. and Oza, N.C. (1999), "Decimated input ensembles for improved generalization", Proceedings of the International Joint conference on Neural Networks, Washington DC
18. Gerra-Salcedo, C. and Whitley, D. (1999), "Genetic approach to feature selection for ensemble creation", Proceedings of Genetic and Evolutionary Computation Conference, pp236-243
19. Optiz, D.W. (1999), "Feature selection for ensembles", Proceedings of 16th International Conference on Artificial Intelligence, pp379-384
20. Therrien, C.W. (1989), Decision estimation and classification: An introduction to pattern recognition and related topics, John Wiley and Sons, NY, 1989.
21. Roli, F., Giacinto, G., and Vernazza, G. (2001), "Methods for designing multiple classifier systems", MCS 2001, pp78-87
22. Opitz, D.W. and Maclin, R.F. (1999), "Popular ensemble methods: An empirical study", Journal of Artificial Intelligence Research, Vol. 11, pp169-198