The Relationship Between Diversity and Accuracy in Multiple Classifier Systems

THESIS

Harris K. Butler, Second Lieutenant, USAF

AFIT-OR-MS-ENS-12-05

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-OR-MS-ENS-12-05

THE RELATIONSHIP BETWEEN DIVERSITY AND ACCURACY IN MULTIPLE CLASSIFIER SYSTEMS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Harris K. Butler, BS

Second Lieutenant, USAF

March 2012

AFIT-OR-MS-ENS-12-05

THE RELATIONSHIP BETWEEN DIVERSITY AND ACCURACY

IN MULTIPLE CLASSIFIER SYSTEMS

Harris K. Butler, BS
Second Lieutenant, USAF

Approved:

//Signed//	14 Mar 2012
Mark A. Friend, PhD (Chairman)	Date
//Signed//	14 Mar 2012
Kenneth W. Bauer, PhD (Member)	Date

AFIT-OR-MS-ENS-12-05

Abstract

A wealth of complementary approaches exists to perform classification of military and non-military items of interest. The goal of fusion techniques is to exploit complementary approaches and merge the information provided by these methods to provide a solution superior than any single method. Associated with choosing a methodology to fuse anomaly detectors and pattern recognition algorithms is the choice of algorithm or algorithms that will be fused. This decision is most often referred to as classifier ensemble selection. Historically classifier ensemble accuracy has been used to accomplish this task. More recently research has focused on creating and evaluating diversity metrics to more effectively select ensemble members. This research focuses on the use of diversity as an ensemble selection methodology and explores the relationship between ensemble accuracy and diversity. Using a wide range of classification data sets, classification methodologies, and fusion techniques it extends current diversity research by expanding classifier domains before employing fusion methodologies. The expansion is made possible with a unique classification score algorithm developed for this purpose. Correlation and linear regression techniques determine the relationship between examined diversity metrics and accuracy is tenuous and optimal ensemble selection should be based on ensemble accuracy.

AFIT-OR-MS-ENS-12-05

To my parents who taught me all the important things in life... To all my classmates who were always ready for a discussion about The Price is Right...

Thank You!

Acknowledgements

I would like to express my appreciation to my thesis committee for their patience and assistance with my research; this has certainly been a learning experience. Also greatly appreciated is everyone who proofread my work and assisted with the writing. Special thanks go to the forklifts of Yellow Trucking for teaching me that real world problems are not at all like case studies from class.

Harris K. Butler

Table of Contents

Page

Page

List of Figures

List of Tables

THE RELATIONSHIP BETWEEN DIVERSITY AND ACCURACY
IN MULTIPLE CLASSIFIER SYSTEMS

I. Introduction

1.1 Background

Relieving the workload of humans is the primary use of computers. Computers are able to perform repetitive tasks much faster than humans and they never get bored. One such task that computers can perform is classifying observations into different classes. The original classification task may have been identifying different species of flower [9], but classification tasks have grown to be more meaningful such as diagnosing tumors [42] [30] and analyzing satellite imagery [29]. Techniques for improving classification accuracy have grown more sophisticated as the field of classification has matured. One such technique to create better classifiers is to combine classifiers into a meta-classifier called an ensemble. By combining multiple classifiers it is possible to create a ensemble that has less error than any of the individual classifiers in the ensemble [31]. There are numerous ways to create classifiers and numerous ways to combine them (called fusion). When deciding which classifiers to fuse one attempts to find a minimum combination of classifiers that produces a robust ensemble. The ideal ensemble would include classifiers that are strong in different areas without overlapping weaknesses. Some researchers have hypothesized the idea of diversity for finding a robust combination of classifiers for fusion. Choosing an ensemble based on diversity may seem like a good idea, but previous efforts do not conclusively show this to be an effective technique.

1.2 Problem Statement

Current research suggests that classifier diversity is of limited use when selecting a classifier ensemble. The reason for this may be the lack of an adequate diversity metric or it may be that no useful relationship exists between diversity and ensemble performance. Previous research has focused on a limited region of classification; focusing primarily on a classification threshold of 50%. We show that even by expanding the space of classification thresholds a relationship between diversity and ensemble performance still does not appear.

1.3 Methodology

Current research focuses on finding an ideal diversity metric that shows a relationship between diversity and ensemble performance. It does not look at how diversity changes as classification thresholds change. A considerable amount of research focuses on how classifier performance changes over the range of classification thresholds [8] [10], but without a corresponding look at the change in diversity the information is of little use for creating classifier ensembles. We show that changing the classification threshold affects the diversity and ensemble performance, but no relationship arises between them that can be exploited for ensemble selection. We also introduce an alternative method for scoring classifier outputs that makes it possible to vary the classification threshold individually for each classifier, which was previously not possible for certain fusion techniques.

1.4 Research Objectives

1. Uncover a relationship between diversity and ensemble performance by using an approach that has never been performed- i.e., varying classification threshold for individual classifiers by use of an alternative scoring technique.

2. If a relationship is discovered, find a way to exploit the relationship to improve ensemble selection.

3. Compare ensemble selection techniques that select classifiers based on accuracy to selection techniques that select classifiers based on diversity.

1.5 Preview

This thesis contains five chapters: an introduction, literature review, methodology, results/analysis, and a conclusion. The introduction introduces the subject matter of the research and provides a framework that the rest of the thesis follows. The literature review contains a primer on relevant classification topics, such as different types of classifiers, measures of classifier performance, types of classifier fusion, and popular diversity metrics. The literature review also contains a review of previous research that looks at the relationship between diversity and ensemble performance. The methodology describes our approach to uncovering a relationship between diversity and ensemble performance. The results/analysis discusses our results and provides evidence for why we believe there is not a meaningful relationship between diversity and ensemble performance. The conclusion discusses the impact of our results, the contribution to diversity studies, and avenues for further research.

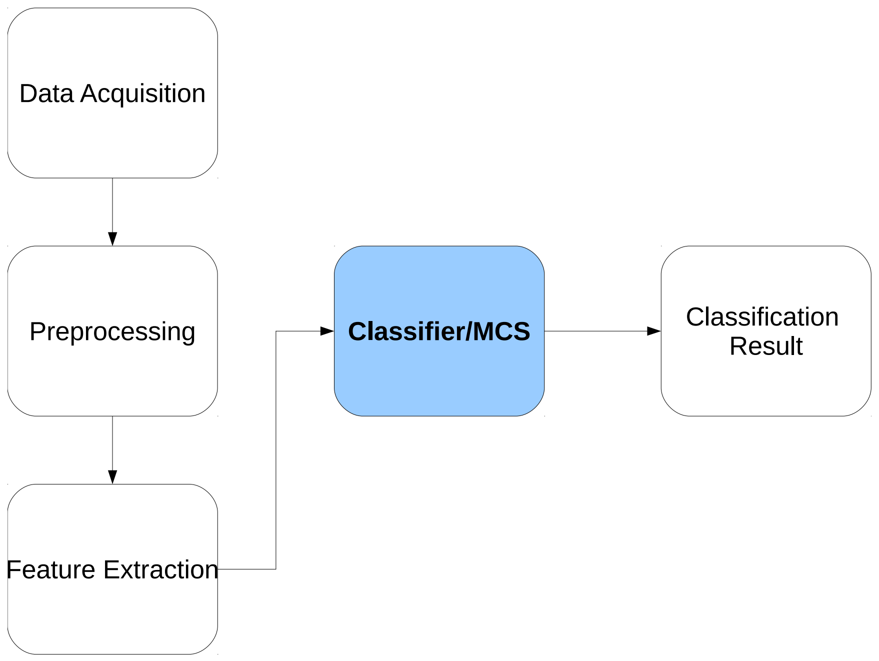
II. Literature Review

This chapter reviews topics relevant to this paper's area of research; it provides a reference for notation and terminology used in later chapters as well as providing some background necessary to understanding the subsequent chapters. This literature review begins with an overview of different classification methods and metrics used to evaluate the performance of individual classifiers. Following the discussion on individual classifiers an overview of the methods used to fuse the output of two or more individual classifiers into a Multiple Classifier System (MCS) as well as how to measure the diversity of the classifiers in the MCS is provided. Methods for creating a set of diverse classifiers will also be discussed. Finally, this chapter concludes with a review of the previous research that investigates the correlation between diversity and accuracy of an MCS.

2.1 Pattern Recognition Process

The pattern recognition process has several discrete steps that hold true for almost every approach. A diagram of the pattern recognition process is shown in figure 1. The process begins with the acquisition of data that characterizes the objects that are to be classified. Data can be gathered by hand, collected with an automated process, or taken from historical records. The quality and accuracy of the data collected is important regardless of the collection method used. Data that is missing values, inaccurate, or inconsistent creates problems that can degrade the accuracy of any classifier, even very robust classifiers. The next step in the process is preprocessing. In this step the data gathered in the previous step is transformed into a usable format. This may involve centering and scaling the data, rotating the data, or deciding what to do with missing values (assigning the average, imputing the values, or removing

Figure 1. The flow through the pattern recognition process



the record). Dimensionality reduction occurs during the feature extraction step. The feature extraction step is the one step in the pattern recognition process that does not always occur. Feature extraction usually occurs with large data sets with possibly redundant information. In this case, it is desirable to reduce the redundancy and dimensionality of the data while still retaining the relevant information. Reducing the data to a useable size is a primary concern in feature extraction, but other goals such as making the features independent can be achieved with some feature extraction algorithms. After the previous steps have been implemented the data is finally ready for classification. Often, multiple classifiers are created and evaluated with the best performing classifier being selected, or possibly multiple classifiers selected to create an MCS. The output of the classifier or MCS must be interpreted to get a final result. Typically the output of a classifier is interpreted and given as either class labels or class probabilities. Once the labels or probabilities are obtained, the pattern recognition process is over.

2.2 Classifiers

In the field of pattern recognition a classifier is an algorithm that attempts to characterize objects of interest into a discrete number of output classes, and can be supervised or unsupervised. Supervised classifiers are trained using a set of data for which the true class is known, while unsupervised classifiers group observations into sets where each group contains observations that are similar to each other and different from the other sets. The research presented in this paper requires data with known classes and employs supervised classifiers. Supervised classifiers provide four categories of output, with the most abstract being a single class prediction assigned to each observation. A slightly more useful output is a ranked list of classes the classifier believes each observation could belong to. This second method can be abstracted to a single class prediction by assigning the highest ranked class to each observation. A third output is a score for each class, which allows for comparing the relative magnitude of each class. This method can be abstracted into a ranked output by ranking each class based on its score. The fourth and most useful classifier output is a list of class probabilities for each observation. Class probabilities are actually a specific version of scoring each class that requires each score fall within the interval $[0, 1]$. By knowing the distribution of the scores provided by a classifier, scores can be transformed into class probabilities.

2.2.1 Classifier Basics.

The research presented in this paper requires the classifiers that provide either scores that may be transformed into probabilities or that natively output class probabilities. The classifiers presented below all meet this criteria. In the event that the classifier only produces scores for each class, enough information is known about the distribution of the scores to transform the scores into class probabilities. Each classifier presented below is popular and well established in the field of pattern recognition.

2.2.1.1 Quadratic Discriminant Analysis.

Quadratic Discriminant Analysis (QDA) classifies observations based on how close they are to the classes in the training set. Classifications with a QDA classifier are made by calculating the Mahalanobis distance (similar to Euclidean distance, but Mahalanobis distance is scale invariant) of the input data to the centroid of each class in the training set, and creating a score using that information; it also accounts for the prior probability of class membership [16]. The formula for QDA is:

$$X_0 \rightarrow \pi_K \text{ if } d_K^Q(X_0) = MAX\left[d_1^Q(X_0), d_2^Q(X_0), ..., d_g^Q(X_0)\right]$$

where:

$$d_i^Q(X_0) = -\frac{1}{2}ln|S_i| - \frac{1}{2}(X_0 - \bar{X}_i)'S_i^{-1}(X_0 - \bar{X}_i) + lnP_i$$

Where the subscript K indicates the assigned class. $d_i^Q(X_0)$ is the score given to observation X_0 for class i, $\bar{X}_i$ is the centroid of class i, S_i is the covariance matrix of class i, and P_i is the prior probability of an observation belonging to class i. $\bar{X}_i$ and S_i are estimated during training. The QDA classifier assumes a multivariate normal population for each class, but does not require that each class have the same variance/covariance structure or that the prior probabilities of each class are equal. Under the assumptions that the variance/covariance structure of each class is the same and the prior probabilities of each class are equal, the method becomes linear and is known as Linear Discriminant Analysis (LDA) [16]. In this paper the assumptions of LDA are not made, and QDA is used. The score outputs from QDA can be transformed into posterior class probabilities using Bayes' Rule.

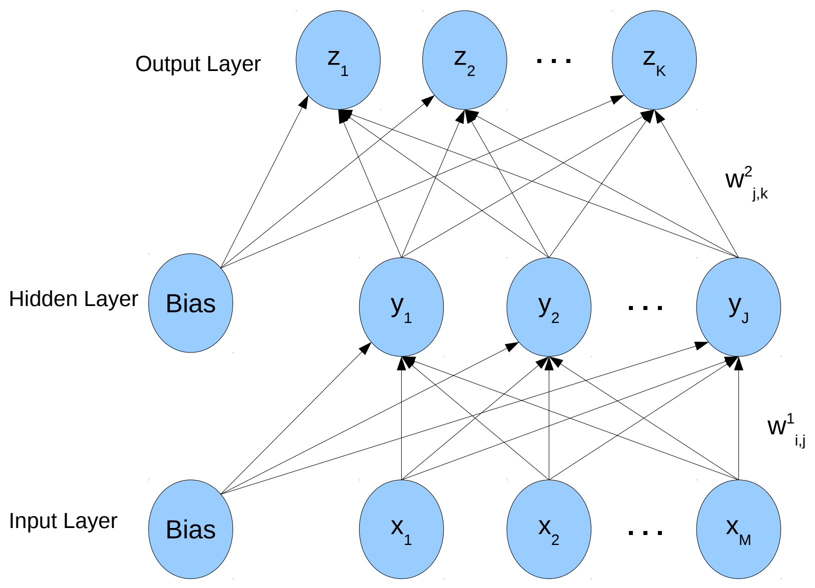
2.2.1.2 K-Nearest Neighbors.

The k-Nearest Neighbors (k-NN) algorithm is one of the simpler classification algorithms, but the classifier is “lazy” in that all computations (i.e., training) are put off until classification time. Because of this training a k-NN classifier takes literally no time at all, but classifying new data points can be computationally intensive. When given an unlabeled observation, the algorithm calculates the Euclidean distance between the unlabeled observation and each of the training data points, and selects the k nearest training data points to the unlabeled data point [16]. Whichever class occurs the most in the k nearest data points is the class that is assigned to the unlabeled data point. The classification calculations become quite computationally intensive as the number of training data points grows large. The k-NN algorithm does not output class probabilities or even scores, but class probabilities can be calculated using information from the k neighbors. Atiya proposes a method of maximum likelihood that performs very well with assigning posterior class probabilities [2].

2.2.1.3 Feed Forward Neural Networks.

In pattern recognition, artificial neural networks (also called neural nets) are a family of classifiers that are approximations to neural networks that occur in biology. Like the name suggests, a neural net is an interconnected network of artificial neurons that apply a simple function to input data and then pass the function value to other neurons in the network. Some people in the pattern recognition community feel that neural nets are over-hyped, but the hype may be because they have been proven to be universal approximators [19]. A typical neural net will have the neurons separated into layers; usually there is one input layer which centers and scales the input data, one or more ”hidden” layers, and an output layer that combines the outputs of the neurons in the last hidden layer into a classification [16]. Each neuron in the hidden layer is usually a simple function, like the sigmoid, which provides outputs to be passed on to the next layer in the net. An aid for visualizing a neural net is shown in figure 2. While it is useful to think of a neural net as a combination of neurons

Figure 2. A top level view of a Neural Network



for training purposes, it is important to remember that a neural net is really a single function, and training the net is only changing the parameters of that function. When data only flows in one direction- forward- then the neural net is called a feed forward neural network. Some more complex neural net designs have multiple hidden layers that may transfer data in more than one direction, these are not feed forward neural networks. Neural nets are sensitive to the number of hidden layers and the number of neurons in each layer- that is to say a poor choice in the size and structure of the neural net can lead to poor classification accuracy. Feed forward neural networks are typically trained through back-propagation, which means that the error of the network is pushed backward through the net and used to adjust the weight that each neuron receives [16]. Feed forward neural networks start with a random set of weights and are trained on one observation at a time. After each observation is fed through the net the weights of each neuron are adjusted to decrease the error. In this manner the neural net hopefully converges on the optimal classifier. The network typically minimizes error through a gradient descent approach with a step size determined by the learning rate, which is decided during the network design. Along with the structure of the net, the learning rate choice can affect how quickly the net converges, or if it converges at all. There are techniques other than gradient descent that can be used and provide faster convergence, but gradient descent is sufficient in most cases. Each time the training set is fed through the net is called an epoch. The network is considered fully trained once it either reaches a predetermined number of epochs or the error has dropped below a predetermined bound. In this research paper one of the neural net algorithms used will be a feed forward neural network with back-propagation (gradient descent). The outputs of a sigmoid function can be treated as native class probabilities since they are on the interval $[0, 1]$.

2.2.1.4 Radial Basis Function Neural Networks.

Radial Basis Function Networks are a special kind of feed forward neural network where the neurons in the hidden layer are radial basis functions [16]. A radial basis function is a function where the value of the function is only dependent on the distance from the origin or some other center point, that is $\theta(x, c) = \theta(||x - c||)$. This differs from the "normal" feed forward neural net which has sigmoid functions as the neurons in the hidden layer. Another difference is that the output neurons are a linear combination of the neurons in the hidden layer. The output of a Radial Basis Function Neural Network can be treated as native class probabilities.

2.2.1.5 Probabilistic Neural Networks.

Probabilistic neural networks are a special kind of radial basis function neural net, and they perform similarly to k-NN classifier. There is one neuron per training observation in the hidden layer, with each neuron containing data on one single training point. When presented with input data, each neuron calculates the distance between the new input data and the neuron's corresponding training observation, which it then passes the calculated distance into the radial basis function [16]. The output layer is a linear combination just like the Radial Basis Function Neural Network, and as such the outputs can also be treated as native class probabilities.

2.2.1.6 Support Vector Machines.

Support Vector Machines are a relatively new family of classifiers that have received great attention in the pattern recognition community. They work by finding a function that most cleanly divides the training data with the maximum distance between the function and the different classes on either side of the function. This distance is also called the margin, so Support Vector Machines are maximum margin classifiers [16]. They attempt to find the best function that divides the classes by performing a kernel trick which performs operations in a space that is of higher dimension than the data, possibly even of infinite dimensions [16]. The key exemplars that are used in finding the dividing function are called support vectors. Support Vector Machines output a score that is not well suited for converting to class probabilities, but can be coerced into class probabilities with some effort.

2.2.2 Classifier Evaluation.

There are multiple ways to evaluate the performance of an individual classifier. Not surprisingly, these ways are typically correlated, but each metric places the emphasis on a slightly different area of performance. This paper will focus primarily on Total Classification Accuracy (TCA) which will be discussed below in the confusion matrix section. Additionally presented are the most common performance metrics which this research also uses to discuss classifier performance when appropriate.

2.2.2.1 Binary Metrics.

With a two class problem, it is convenient to associate one of the classes as a *positive* and one as a *negative*. With this terminology, there are four possible instances. When the observation is actually in the positive class and the classifier correctly labels it, then it is a *true positive* (tp). Similarly, when the observation is actually in the negative class and the classifier correctly labels it, then it is a *true negative* (tn). When the classifier incorrectly labels an observation, then it is a *false positive* (fp) if the true class is negative, and a *false negative* (fn) if the true class is positive. When evaluating a classifier, it is useful to keep track of the number of times each of these four instances occur. By knowing each occurrence of the four possible instances, it is possible to calculate a number of useful metrics that can be used to compare the performance of different classifiers. The true positive rate, also called hit rate or recall, is the number of true positives divided by the total number of positives; it is desirable to have a high true positive rate [10]:

$$True\ Positive\ Rate = \frac{tp}{tp + fn}$$

The false positive rate is the number of false positives divided by the total number of negatives; it is desirable to have a low false positive rate [10]:

$$False\ Positive\ Rate = \frac{fp}{fp + tn}$$

The precision of a classifier is defined as the number of true positives divided by the total number of observations that the classifier identified as positive; it is desirable for a classifier to have a high precision [10]:

$$Precision = \frac{tp}{tp + fp}$$

The accuracy of a classifier is the number of observations correctly classified divided by the total number of observations; it is desirable for a classifier to have a high accuracy [10]:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

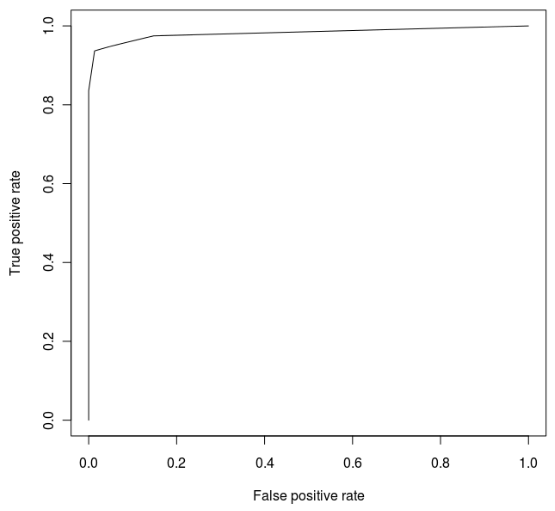
Finally, another measure of classifier performance is the F-measure, which is two divided by the reciprocal of the precision plus the reciprocal of the true positive rate; it is desirable for this metric to be high [8]:

$$F - measure = \frac{2}{\frac{1}{Precision} + \frac{1}{True\ Positive\ Rate}}$$

2.2.2.2 ROC Analysis.

Another common method of evaluating classifiers is by comparing their Receiver Operating Characteristic Curves (ROC curves). ROC curves are most easily interpreted in two class problems however higher dimensional versions exist. ROC curves display the trade off between true positives and false positives [10]. If a classifier only outputs class labels and not class probabilities, then it only exists as a single point on the ROC space, and does not have a ROC curve. Classifiers that output scores or class probabilities can vary the threshold at which they make assignments, and this is how a ROC curve is generated. By overlaying two or more ROC curves on the same plot, a visual comparison can be made of different classifiers. It is desirable to have a ROC curve occupy higher places on the plot, as a high point indicates a high true positive rate at the selected false positive rate. An example plot showing a ROC curve is shown in Figure 3. Like many metrics, it is possible to calculate confidence regions for ROC curves [27], although this is of limited use with the research in this thesis. Also possible is using ROC curves to combine multiple classifier outputs [17] into a more robust single output, however this option is not pursued in this research. Knowing that points near the upper left corner of the plot are best, it

Figure 3. An example of a ROC curve



is possible to create a numerical metric associated with the visual comparison. The Area Under the ROC Curve (AUC) is easy to interpret, and a high AUC indicates a classifier that performs well. An AUC of 1 indicates a perfect classifier, while an AUC of 0.5 indicates a classifier that performs no better than random chance [8]. AUC's of less than 0.5 indicate that the classifier gives the wrong classification more often than not, and using the complement of the classifier output would yield better performance. The AUC is equal to the Wilcoxon-Mann-Whitney test statistic and is the probability that a randomly selected positive observation will be scored higher (by the classifier) than a randomly selected negative observation. A similar metric to AUC is also sometimes used, called the Gini Coefficient. The Gini Coefficient comes from the field of economics, and is related to the AUC by the relationship: $G = 2AUC - 1$ [8]. While AUC takes on values from $[0, 1]$, the Gini Coefficient takes on values from $[-1, 1]$ which is somewhat more intuitive to interpret, since it means a classifier that performs no better than random chance has a Gini Coefficient of 0. Positive Gini coefficients indicate a high performing classifier while negative coefficients indicate a low performing classifier. A perfect classifier would have a Gini Coefficient of 1. Some other commonly used terms associated with ROC curves are the sensitivity, which is equal to the true positive rate, and the specificity, which is: $Specificity = 1 - False\ Positive\ Rate$. There is also the positive predictive value, which is equal to the precision [8].

2.2.2.3 Confusion Matrix/Total Classification Accuracy.

The ideas of the previously mentioned binary metrics and ROC analysis are really only useful with binary classification problems but there are many cases where there are more than two classes. In these other cases it is possible to present the information in a confusion matrix. A confusion matrix has the actual classes comprising the rows of the matrix and the predicted classes comprising the columns of the matrix. If a classifier correctly identifies an observation it will fall somewhere on the main diagonal of the confusion matrix, while misclassification will fall somewhere off the main diagonal. Since correct classifications are conveniently located on the main diagonal of the confusion matrix, the Total Classification Accuracy (TCA) can be defined as the trace of the confusion matrix divided by the total number of observations. An example of a confusion matrix and its TCA are shown in table 1:

Table 1. Example of Confusion Matrix and TCA

			Predicted	
		Class 1	Class 2	Class 3
	Class 1	8	1	0
Actual	Class 2	2	9	1
	Class 3	1	6	9

$$TCA = \frac{8+9+9}{8+1+0+2+9+1+1+6+9} = \frac{26}{37} = 70.27\%$$

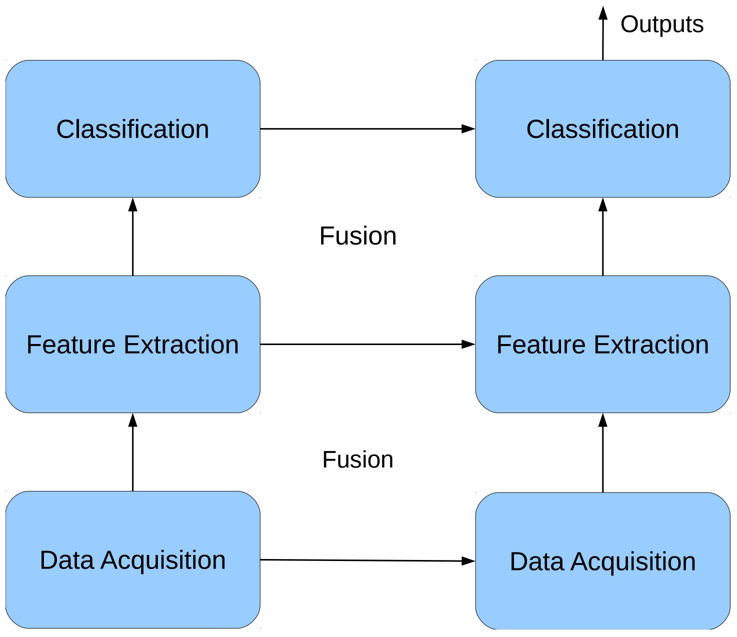
2.3 Fusion

One way to achieve better accuracy is to combine information at some point in the decision process. Combining multiple sources of information is called fusion. Fusion can happen at multiple levels, the levels are usually separated by what type of information they are combining. Thus, data level fusion would combine raw data, feature level fusion combines the information after it has been processed into features, and classifier fusion combines the outputs of individual classifiers. There is general consensus among researchers that maximum benefit occurs when fusing information at the lowest possible level, the data level, but this is not always possible and the higher levels of fusion are typically much easier to perform. The research in this thesis focuses on classifier level fusion and works with data that has already been through the pre-processing and feature extraction steps, so data and feature level fusion is not possible. Figure 4 shows a visual representation of information fusion.

2.4 Classifier Fusion

While it is certainly possible to classify observations with a single classifier, greater accuracy may achieved by creating multiple classifiers and combining the results. The underlying principle behind this idea is that classifiers can be strong in different areas

Figure 4. Different Levels of Fusion

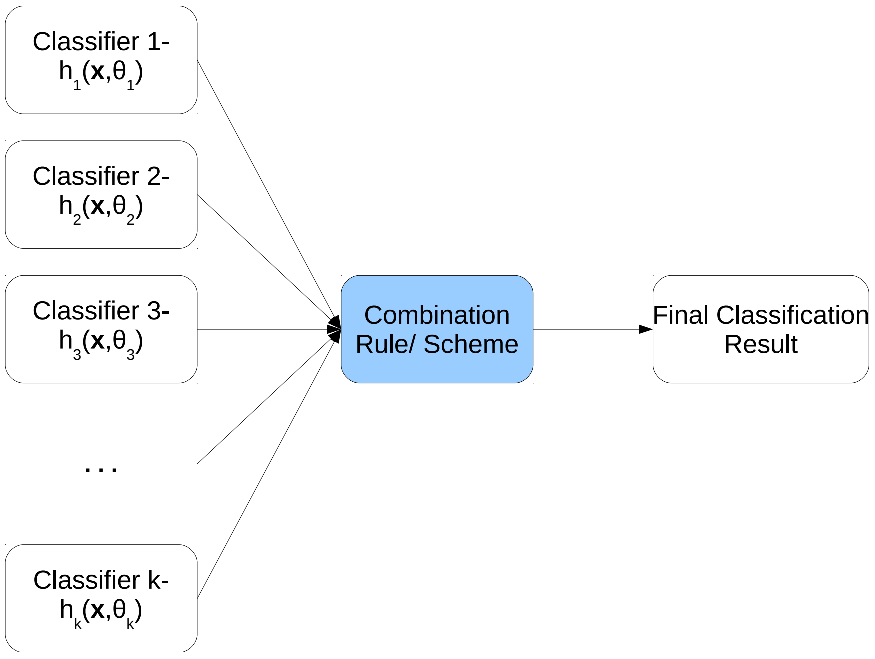


of the decision space, and greater accuracy can be achieved by combining classifiers with different strengths. Combining multiple classifiers creates a Multiple Classifier System (MCS), and there are many different combination rules. However the classifiers are combined, the basic structure of an MCS is the same, shown in figure 5. The combination rule may not necessarily accept classifier outputs in a parallel manner, it may accept the individual classifier outputs in a hierarchical/non-parallel or even serial order. Fundamentally, all rules fall into three different levels; there is the abstract level which only requires class labels as outputs, there is the rank level which requires a ranked list of class outputs, and finally there is the measurement level which requires class probabilities.

2.4.1 Abstract Level Fusion.

Abstract level classifier fusion is the simplest way to combine classifiers, and it requires the least amount of information. At this level, only class labels are required

Figure 5. The structure of an MCS


Classifier 1- $h_1(\mathbf{x},\theta_1)$
Classifier 2- $h_2(\mathbf{x},\theta_2)$
Classifier 3- $h_3(\mathbf{x},\theta_3)$
...
Classifier k- $h_k(\mathbf{x},\theta_k)$
Combination Rule/ Scheme
Final Classification Result

and the combination rules are computationally easy. There are numerous abstract level fusion methods, only a few of the most popular are discussed here.

2.4.1.1 Majority Vote.

Majority voting is the simplest and most intuitive way to fuse multiple classifiers. It involves selecting the most commonly assigned class as the final assigned class. It requires at least two classifiers make an assignment to at least one class. If there is a case where no class gets more than one assignment, the final assignment is given to the individual classifier with the best accuracy [32]. There are other frameworks for majority voting than the simple majority described above; there is unanimous voting where all classifiers must agree to make an assignment, plurality voting where at least 50% of the classifiers must agree, or variable threshold voting where a class is not assigned unless the number of votes for a class is above a predetermined threshold [32].

2.4.1.2 Weighted Majority Vote.

Weighted majority voting is similar to majority voting, except each classifier in the MCS is given a certain weight, with the weight typically corresponding to the accuracy of the classifier [32]. This allows the more accurate classifiers to have more "say" in the voting, but if a number of weaker classifiers vote for a class they can overturn the vote of the stronger classifier. Weights are generally selected so they sum to one.

2.4.1.3 Behavior Knowledge Space.

Behavior Knowledge Space (BKS) is an abstract level fusion scheme but is more complicated than voting. During training a lookup table is created that contains every possible classifier output combination [32]. The classifier output combinations from the training set are used to estimate truth values and their relative frequencies. When a new exemplar is classified the classifier output combination for that exemplar is found in the lookup table, and assigned the "Truth" value with a confidence level equal to the relative frequency. Table 2 is an example BKS lookup table with a two class, two classifier problem: Table 2 shows both the strength and weakness

Table 2. Example BKS table

Classifier 1	Classifier 2	Truth	(% occurrence)
0	0	0	(76%)
0	1	1	(75%)
1	0	0	(51%)
1	1	1	(85%)

of BKS fusion. In the example, Classifier 2 is very good at identifying observations with class 1, but has a high false negative rate. Classifier 1 is not particularly strong in identifying observations with either class. Therefore, when Classifier 2 assigns a label of class 1 it can be said with reasonable certainty that the true class is 1. Also, when both classifiers agree that an observation is in class 0, they are probably correct. However, when classifier 1 assigns a label of class 1 and classifier 2 assigns a label of class 0, then an ambiguous result happens. While BKS can allow for greater flexibility when classifiers disagree, it may sometimes produce results where the confidence level is not very high.

2.4.2 Rank Level Fusion.

The next level of classifier fusion is rank level fusion. To use a rank level fusion technique more information than just a single class label is required. Either scores that can be sorted into a ranked list or a classifier that outputs a ranked list of classes is necessary. While the main drawback is that rank level fusion requires more information, the hope is that rank level fusion is more accurate than abstract level fusion because it takes into account the extra information. Rank level fusion is sometimes used to reduce the number of possible classes while hopefully retaining the true class as a possibility. These methods, called Class Set Reduction methods, are not discussed here because they do not provide a single class label as a final output. Rank level fusion is also used to reorder the class set in the hope of getting the true class to the top rank. Two of these methods, called Class Set Reordering methods, are discussed below.

2.4.2.1 Borda Count.

Borda Count is a rank level fusion method that is similar to majority voting, but it does not discard a classifier's support for the lower ranked classes. With m classes, the top ranked class from a classifier receives $m\text{-}1$ votes, the second ranked class receives $m\text{-}2$ votes, all the way down to the m_{th} ranked class receiving zero votes [37].

The votes are then added up and the class with the most votes is the class that is assigned.

2.4.2.2 Logistic Regression.

The Borda Count method does not take into account the relative quality of each classifier. The Borda Count method can therefore be improved by assigning weights to each classifier relative to its performance and performing a logistic regression to estimate new values for each class [37]. These new values can then be used to make a class assignment.

2.4.3 Measurement Level Fusion.

Finally, measurement level fusion requires even more information than rank level fusion, but again the hope is that measurement level fusion schemes perform even better due to the additional information. Measurement level fusion schemes require fuzzy measures on the interval $[0, 1]$ as the classifier outputs which are treated as class probabilities or one of the other measures of evidence: possibility, necessity, belief, or plausibility. There are very many measurement level fusion schemes, and only some of the most popular are discussed below. With measurement level fusion, the following symbol conventions are used:

- $\mu_j(\mathbf{x})$- the support given by the MCS to class j for an observation $\mathbf{x}$. For example, if an MCS believes that exemplar $\mathbf{x}$ belongs to class j with probability 0.95, then $\mu_j(\mathbf{x}) = 0.95$
- $d_{t,j}(\mathbf{x})$- the support given by the individual classifier t to class j for an observation $\mathbf{x}$. This is similar to $\mu_j(\mathbf{x})$, but $d_{t,j}(\mathbf{x})$ is the support from an individual classifier and not the entire system of classifiers.
- T- the number of classifiers in the MCS

2.4.3.1 Generalized Mean.

The generalized mean fusion method encompasses many commonly used fusion methods. The formula for a generalized mean fusion is [32]:

$$\mu_j(\mathbf{x}, \alpha) = \left(\frac{1}{T} \sum_{t=1}^{T} d_{t,j}(\mathbf{x})^{\alpha} \right)^{1/\alpha}$$

The choice of α determines the behavior of the rule. If $\alpha = 1$ we obtain the mean rule, also called the basic ensemble model (BEM),:

$$\mu_j(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} d_{t,j}(\mathbf{x})$$

If $\alpha = -\infty$ then we obtain the minimum rule:

$$\mu_j(\mathbf{x}) = \min_{t=1 \ldots T} \{d_{t,j}(\mathbf{x})\}$$

Similarly, if $\alpha = \infty$ then we obtain the maximum rule:

$$\mu_j(\mathbf{x}) = \max_{t=1 \ldots T} \{d_{t,j}(\mathbf{x})\}$$

When $\alpha = 0$ we obtain the geometric mean, which is a modified form of the product rule (discussed later):

$$\mu_j(\mathbf{x}) = \left(\prod_{t=1}^{T} d_{t,j}(\mathbf{x}) \right)^{1/T}$$

2.4.3.2 Trimmed Mean.

The trimmed mean rule is a way of avoiding instances where an individual classifier gives unusually high or low support to a particular class. For a P% trimmed mean, the top and bottom P% classifiers are removed from the MCS for that observation, and the mean rule is applied to the $1 - 2P\%$ in the middle. This avoids instances where one "rogue" classifier gives an level of support that may shift the mean [32].

2.4.3.3 Median Rule.

The median rule is a statistical rule similar to the minimum or maximum rule, but unlike the minimum or maximum rules it does not belong to the generalized mean family of fusion rules [32]. The median rule selects the median level of support:

$$\mu_j(\mathbf{x}) = \underset{t=1...T}{\text{median}}\{d_{t,j}(\mathbf{x})\}$$

2.4.3.4 Product Rule.

The product rule multiplies the support given by each classifier and if the posterior probabilities are correctly estimated then the product rule gives the best estimate of the overall class probabilities [32]. However, if one classifier gives very low support to a class it effectively removes the chance of that class being selected:

$$\mu_j(\mathbf{x}) = \frac{1}{T}\prod_{t=1}^{T} d_{t,j}(\mathbf{x})$$

2.4.3.5 Generalized Ensemble Model.

The Generalized Ensemble Model (GEM) is a generalized model of the mean rule, also called the Basic Ensemble Model (BEM). At its core, GEM is a weighted average of the support given by each classifier:

$$\mu_j(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t d_{t,j}$$

The $\alpha's$ are selected in a way that minimizes the mean squared error of the MCS. This is done by calculating the *misfit function* for each classifier:

$$m_i(x) = f(x) - f_i(x)$$

Where $f(x)$ is the truth and $f_i(x)$ is the output of classifier i. The correlation matrix between the misfit functions of all the classifiers is then constructed, with individual entries:

$$C_{ij} = E\left[m_i(x)m_j(x)\right]$$

The weights, α_i, are calculated using the entries in the correlation matrix:

$$\alpha_i = \frac{\Sigma_j C_{ij}^{-1}}{\Sigma_k \Sigma_j C_{kj}^{-1}}$$

Perrone gives a proof that shows that the weights calculated in this way will give the linear combination of classifiers that minimizes the MSE. It performs better than the best individual classifier and better than BEM. For more information on GEM, see [31].

2.4.3.6 Decision Templates.

Kuncheva proposes a method of creating decision templates that are the average decision profile observed for each class in the training set. The final support for a given observation is then calculated using some similarity metric between the observation's decision profile and the decision templates for each class. The similarity metric used is typically a squared Euclidean distance, but it could be any suitable measure. For more information on Decision Templates, see [22].

2.4.3.7 Dempster-Schafer Based Combination.

Dempster-Schaefer Based Combination makes use of belief functions, and treats classifier outputs as evidence provided by a source (trained classifier) [37]. The evidence is transformed into belief values, and Dempster's combination rule is applied to the belief values. Dempster's combination rule states the belief values from each source should be multiplied to find the final support given to a class. For more information on Dempster-Schaefer Based Combination see [26] or [35].

2.5 Diversity

The entire point of fusing multiple classifiers together is to balance the weaknesses of each individual classifier. This requires classifiers that make errors in different areas of the decision space. Classifiers that are strong in different areas are said to be diverse. To create effective MCSs we must fuse classifiers that are diverse because fusing non-diverse classifiers provides no benefit. In the sections below some measures of diversity are discussed, how to create a diverse set of classifiers, as well as a review of the studies that have attempted to uncover the relationship between diversity and the performance of an MCS. It is worth noting that while there may be a relationship between diversity and accuracy in practical scenarios, Kuncheva and Kounchev show a method of how to create classifiers that target a specific accuracy and diversity [20]. This means that for every diversity metric here there is a way to construct classifier outputs to achieve the same ensemble accuracy with two different diversity measures. For each diversity metric discussed an example is given of how classifier outputs may have the same diversity but vastly different accuracies.

2.5.1 Diversity Metrics.

Diversity is easy to understand intuitively, but difficult to quantify. Because of the disconnect between understanding diversity and quantifying it, there are many different measures that have been proposed as a measure of diversity. Some of the most popular metrics are discussed below. Most diversity metrics are designed for pairwise comparisons of classifiers. There are a few that can handle more than two classifiers, but it is also common to compare multiple classifiers with pairwise diversity metrics by computing the diversity of every pairwise combination and averaging these results. In the pairwise diversity metrics, the convention used is the letters a, b, c, d represent fractions of instances where a is the fraction that are correctly classified by both classifiers, b is the fraction that is correctly classified by classifier i but misclassified by classifier j, c is the fraction that is correctly classified by classifier j but misclassified by classifier i, and d is the fraction that is misclassified by both classifiers as shown in table 3.

Table 3. Reference for Pairwise Diversity Metrics

	Classifier j is correct	Classifier j is incorrect
Classifier i is correct	a	b
Classifier i is incorrect	c	d

2.5.1.1 Correlation.

One of the most commonly used diversity metrics is the correlation between two classifiers, $\rho_{i,j}$ [23]. Maximum diversity is obtained when $\rho_{i,j} = 0$, indicating two completely uncorrelated classifiers. $\rho_{i,j}$ is calculated as:

$$\rho_{i,j} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}},\ 0 \leq \rho_{i,j} \leq 1$$

Two identical classifiers that always label exemplars the same have $\rho = 1$ and fusing them with BEM will give an MCS that has a TCA equal to the accuracy of the individual classifiers. Another set of identical classifiers will also have $\rho = 1$, but if the accuracy of the individual classifiers in this new set does not equal the accuracy in the previously mentioned classifiers, then they will not have the same TCA.

2.5.1.2 Yule's Q.

Yule's Q statistic, $Q_{i,j}$ is another commonly used diversity metric, and takes on positive values if both classifiers tend to correctly classify the same instances, and negative values if both classifiers tend to incorrectly classify the same instances [44]. Maximum diversity is achieved at $Q_{i,j} = 0$. $Q_{i,j}$ is calculated as:

$$Q_{i,j} = \frac{ad - bc}{ad + bc}$$

Two different MCSs can have the same Yule's Q statistic as long as the products ad and bc remain the same. For example, if one MCS has $a = 0.85$, $b = 0.05$, $c = 0.05$, and $d = 0.05$ and the other classifier has the same values except a and d have swapped values so $a = 0.05$ and $b = 0.05$ then both MCSs will have the same Yule's Q statistic but the first MCS will be very strong and the second MCS will be very weak.

2.5.1.3 Disagreement.

Disagreement, $D_{i,j}$, is the probability that the classifiers will disagree, and is calculated as [23]:

$$D_{i,j} = b + c$$

Maximum diversity is achieved when $D_{i,j} = 1$. Two different MCSs can have the same disagreement but different accuracies as long as the sum $b+c$ remains the same.

Like the example given for Yule's Q statistic, if the values a and d swap values one MCS will be strong while the other MCS will be weak even though they have the same disagreement.

2.5.1.4 Double Fault.

Double fault, $DF_{i,j}$ is the probability that both classifiers will misclassify an observation, and is equal to d [23]:

$$DF_{i,j} = d$$

Maximum diversity is achieved when $DF_{i,j} = 0$. It is worth noting that the Double Fault metric only considers when two classifiers are non-diverse in a *bad* way, i.e., they are both wrong. When both classifiers are correct the Double Fault metric does not decrease. Two MCSs will have the same double fault value as long as they have equal values d. One MCS may have 99% "double correctness" and 1% double faults, while another MCS may have 99% "single faults" and 1% double faults. The former MCS is far more robust than the latter MCS despite them having the same double fault values.

2.5.1.5 Entropy.

Entropy, E, operates under the assumption that diversity is highest if half of the classifiers are correct and half of the classifiers are wrong. Diversity is highest when $E = 1$ and lowest when $E = 0$. Entropy is calculated as [32]:

$$E = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T - \lceil T/2 \rceil} \min\{\zeta_i, (T - \zeta_i)\}$$

Where T is the total number of classifiers, ζ_i is the number of classifiers that misclassified the observation $\mathbf{x}_i$, therefore $(T - \zeta_i)$ is the number of classifiers that correctly classifierd observation $\mathbf{x}$, and N is the number of observations in the data set. These definitions will also be used in the formula for Kohavi-Wolpert Variance, discussed below. Two MCSs can have the same entropy values but different accuracies. For example, if one MCS always has three correct classifiers and two incorrect classifiers and the second MCS always has two correct classifiers but three incorrect classifiers then they will the have the same entropy values but different accuracies.

2.5.1.6 Kohavi-Wolpert Variance.

Kohavi-Wolpert Variance is similar to the disagreement measure but can be calculated with more than two classifiers. Diversity is maximized when Kohavi-Wolpert Variance is high. Kohavi-Wolpert Variance is calculated as [23]:

$$KW = \frac{1}{NT^2} \sum_{i=1}^{N} \zeta_i (T - \zeta_i)$$

Kuncheva has proven that Kohavi-Wolpert Variance of an MCS is related to the average of all pairwise disagreement measure in the MCS by the relationship [23]:

$$KW = T * \frac{\sum_{i=1}^{T-1} \sum_{j=i+1}^{T} D_{i,j}}{\binom{T}{2}}$$

Kohavi-Wolpert Variance can be effected the same way as the entropy measure. Two MCSs can have the same KWV values but different accuracies. For example, if one MCS always has three correct classifiers and two incorrect classifiers and the second MCS always has two correct classifiers but three incorrect classifiers then they will the have the same KWV values but different accuracies.

2.5.1.7 Difficulty.

The measure of difficulty, θ, uses a random variable Z defined as the fraction of classifiers in the MCS that misclassify an observation $\mathbf{x}_i$. Therefore $Z(\mathbf{x}_i)$ can take on values $\{0, 1/T, 2/T, 3/T, ..., 1\}$. The measure of difficulty is defined as $\theta = var(Z)$, and can be estimated with the sample variance of the *Z's* from the training set [23]. Diversity is maximized when θ is high. Two classifiers can have the same difficulty value when they misclassify the same observations but with different class probabilities. A classifier may misclassify an observation by assigning a class probability of 0 or anything up to but not including 0.5 if the classification threshold is 0.5. However, misclassifying an observation with a class probability of 0 "hurts" most measurement level fusion methods much more than misclassifying with a class probability of 0.49. An MCS that has a lot of misclassifications with class probabilities of 0 will have the same difficulty measure as an MCS that has the same amount of misclassifications except with class probabilities of 0.49 although their accuracies will likely be very different. The first MCSs misclassifications are very far off target, while the second MCSs misclassifications are near misses.

2.5.2 Creating Diversity.

It is easiest to create a diverse MCS when there are a large number of classifiers to choose from. Intuitively, having few classifiers to choose from limits the choices, while having many classifiers allows for picking and choosing the most diverse set of classifiers. There are many ways to create multiple classifiers from just one set of data. Some of the most popular ways to do so are discussed below.

2.5.2.1 Injecting Randomness.

Some classifiers have random starting conditions, such as the weights in a neural network. With these types of classifiers, multiple classifiers can be created by training several instances of the same classifier type on the same training set but using different starting weights [36]. Given the different starting weights, each classifier should end up finding slightly different decision boundaries.

2.5.2.2 Data Splitting.

A simple way to construct multiple classifiers is to split the training data into N disjointed subsets, which can then be used to train N classifiers. One of the disadvantages of this technique is that it cannot be used on small training sets where splitting the data would lose too much information [36].

2.5.2.3 Cross Validation.

Cross validation is a method to construct multiple classifiers that does suffer the weakness of simple data splitting. The training data is split into N subsets like data splitting, but they are constructed differently. The difference between data splitting and cross validating is that the actual N cross validation sets are overlapping $N-1$ subsets, with each training set leaving out a different subset. This method therefore retains more information in the training sets compared to the data splitting method, with the drawback that much of the information overlaps [36].

2.5.2.4 Bagging.

Perhaps the best method of creating many classifiers is bootstrap aggregating, also known as bagging. It involves constructing multiple training sets out of one original training set by creating several samples of the same size as the original training set by using a statistical technique known as bootstrapping. Bootstrapping involves creating a sample the same size as the original data set by pulling n individual samples with replacement from the training set, where n is the size of the training set. Each entry in the training set may appear in the bootstrapped sample anywhere from zero to n times. Each bootstrapped sample usually contains small changes with respect to the original training sample. The resulting bootstrapped samples are each used to train a classifier [36]. If the classifier is unstable, the resulting trained classifiers on each sample will show considerable differences, but combining them into an MCS will reduce the variance.

2.5.2.5 Boosting.

Boosting is an iterative technique used to create a strong classifier that is a combination of weak classifiers. There are many different boosting algorithms, but each follows the same basic steps. Each iteration trains a classifier on the set of observations that the previously trained classifier misclassified. While boosting is certainly powerful, it is an ensemble learning technique and not suitable for creating multiple classifiers to be used in other fusion methods [36].

2.5.2.6 Feature Selection.

Another way to create multiple classifiers is to create training sets that contain feature subsets of the original feature set. The feature subsets can be generated randomly, or they can be generated intelligently by grouping complementary features together. This method works well when the data is of high dimensionality or has many redundant features [36].

2.5.2.7 Noise Injection.

A training set can be modified by adding in a small amount of noise to some of the features in the data. The noise should have zero mean and a small covariance. By adding in random noise to a training set, several training sets can be created that are different from the original training set, while containing the same information on average [36].

2.5.3 Prior research.

The intuition is that a diverse set of classifiers should perform better and deliver better accuracy than any single classifier in the MCS. If this is true, there should be some positive correlation between the diversity of an MCS and its accuracy. If there is a correlation, there should be a way to select which classifiers are included in an MCS by using diversity as a criteria. Multiple studies have been performed investigating this relationship. Aksela and Laaksonen study classifier selection using a number of diversity metrics and fusion techniques and state that diversity metrics that disregard classifier correctness are not optimal for selection purposes [1]. However, diversity metrics that take classifier correctness into account are "cheating" by really making the measure about accuracy instead of diversity. Aksela and Laaksonen also state that it is desirable for the diversity of the errors to be high, but the agreement on the correct outputs should also be high [1]. This statement of diversity being important but not at the cost of accuracy is echoed in other research as well. Brown and Kuncheva decompose their diversity into "good" and "bad" diversity where increasing good diversity reduces error and increasing bad diversity increases error [3]. However, the popular diversity metrics in use today are not decomposed into good and bad diversity, and a separate decomposition must be performed/derived for every combination of loss function and fusion method [3]. Brown and Kuncheva also did not provide a way to use the good/bad diversity decomposition for building classifier ensembles. Canuto *et al.* perform a study on ensemble selection with both hybrid (different types of classifiers) and non-hybrid (all classifiers are the same type) ensembles. They determined that classifier selection does have an impact on an ensemble's accuracy and diversity but they do not show any link between accuracy and diversity [4]. They also show that hybrid ensembles provide the most diversity, this is one reason we use hybrid ensembles in this research. Gacquer *et al.* propose a genetic algorithm for ensemble selection that performs well with a specified accuracy-diversity trade off of 80/20, indicating that diversity must be of at least some use for selecting ensembles that generalize well over a population [13]. However they mention that this may not be true for small data sets, and it may not be true even for all large data sets as well. Hadjitodorov *et al.* look at cluster ensembles which is a non-supervised learning technique, but still offer valid insight. They claim that accuracy peaks somewhere around medium diversity, and very high or very low diversity ensembles are a poor choice [15]. Kuncheva, who has performed a great deal of research in the area, states that while no relationship between diversity and accuracy has been conclusively proven it is may still be a useful idea in creating ensemble selection heuristics [21]. Kuncheva and Whitaker note that the diversity metrics tend to cluster with themselves indicating that there is some agreed upon idea of diversity, but state that using diversity for enhancing the design of ensembles is still an open question [23]. Ruta and Gabrys show a correlation between one measure of accuracy, majority voting error, and two diversity metrics, the pairwise double-fault measure and the non-pairwise fault majority measure, but this correlation is limited to just that single fusion method and those two diversity metrics [38]. In addition, the non-pairwise fault majority measure of diversity was designed specifically for majority voting fusion, and thus is expected to show a relationship with majority voting error [38]. Shipp and

Kuncheva consider a large number of diversity metrics and fusion methods and have interesting findings but have “discouraging” results with relation to the correlation between ensemble accuracy and diversity– they did not find one [39]. Windeatt proposes a diversity metric that is measured across classes and not classifiers; he shows it to be correlated with the base classifier’s accuracy but it does not appear to be correlated with the accuracy of the MCS as a whole [44]. All of the studies above use a variety of classifiers, fusion methods, and diversity metrics. Tables 4, 5, and 6 show which classifiers, fusion methods, and diversity metrics were used in each study including this thesis research.

Table 4. Classifiers used in prior research

Authors	Classifiers										
	MLP	SVM	Linear	k-NN	Fuzzy-MLP	RBFN	Tree	JRIP	Cluster	Quad	PNN
Aksela and Laaksonen [1]	x	x									
Brown and Kuncheva [3]			x								
Butler and Friend	x	x		x		x				x	x
Canuto et al. [4]	x	x		x	x	x	x	x			
Gacquer et al. [13]							x				
Hadjitodorov et al. [15]									x		
Kuncheva and Whitaker [23]			x							x	
Ruta and Gabrys [38]			x							x	
Shipp and Kuncheva [39]			x							x	
Windeatt [44]	x										

Table 5. Fusion methods used in prior research

Authors	Fusion methods													
	Maj Vote	BKS	DEC	Naive Bayes	Sum	BEM	Median	MLP	Fuzzy-MLP	Single link	WER	Max/Min	Product	DT
Aksela and Laaksonen [1]	x	x	x											
Brown and Kuncheva [3]	x													
Butler and Friend	x				x	x						x	x	
Canuto et al. [4]														
Gacquer et al. [13]	x													
Hadjitodorov et al. [15]	x		x	x	x	x	x	x	x					
Kuncheva and Whitaker [23]	x	x		x		x				x	x	x	x	x
Ruta and Gabrys [38]	x													
Shipp and Kuncheva [39]	x	x				x					x	x	x	x
Windeatt [44]	x				x									

Table 6. Diversity metrics used in prior research

Authors	Diversity metrics																	
	KW Variance	Mutual information	Correlation	Yule's Q	Disagreement	Double fault	Mutual information of errors	K-L divergence	Entropy	Kappa	Rand index	Spectral	Difficulty	GD	CFD	Margin	Fault Majority	Sharkey's r
Aksela and Laaksonen [1]	x	x	x	x	x	x	x											
Brown and Kuncheva [3]								x										
Butler and Friend	x		x	x	x	x			x									
Canuto et al. [4]						x			x									
Gacquer et al. [13]										x								
Hadjitodorov et al. [15]								x		x								
Kuncheva and Whitaker [23]	x		x	x	x	x		x	x			x	x	x				
Ruta and Gabrys [38]	x		x	x	x	x			x	x		x				x	x	
Shipp and Kuncheva [39]	x		x	x	x	x			x	x		x	x	x				
Windeatt [44]			x	x	x	x						x			x			

While some of the studies claim a correlation between accuracy and a proposed diversity measure, all of the studies fall short of conclusively proving a link between diversity and accuracy. Part of the problem stems from the fact that there is no formal definition of diversity, however, there are a myriad of options to measure it. Kuncheva goes so far as to say that there is probably no single diversity metric that is consistently correlated with accuracy; this is reminiscent of Wolpert's No Free Lunch Theorems (for supervised learning) that state there is no single optimal classification method [21] [45]. With the current state of research in this area, it is safe to say that the issue still has not been decided and there is still insight to be gained. One area that has not been researched at all is what happens when the classification threshold is changed. All previous studies focused on the correlation between the classification accuracy at a fixed classification threshold- i.e., for a two class problem, the class with posterior probability greater than 0.5 was used. Hopefully more light can be shed on the relationship by examining what happens to diversity and accuracy as the classification threshold is changed. By approaching accuracy and diversity in this new direction it may be possible to create another measure of diversity that can be used in selecting classifiers for an MCS.

2.6 Data sets

Fourteen different data sets are used for the research in this paper. All data sets are available from the UCI Machine Learning repository [12]. While some of the information in the data sets may seem to be of a private nature, all personally identifiable information has been removed from the data. A brief description of each data set is given below.

2.6.1 Balance Scale.

The goal of this data set is to predict whether a scale will be balanced, given the weight on each side and the distance of the weight on each side from the center of the scale. While this is an easy task for a human, the fact that the "unbalanced" class occupies space to the left and right of the "balanced" class makes it an interesting problem for classifiers [40]. There are 625 observations and 4 numerical features.

2.6.2 Breast Cancer Wisconsin.

The goal of this data set is to predict whether a tumor is malignant or benign based on the measurements of the cells of the tumor [30]. There are 699 observations, 9 numerical features, and 1 sample ID number which is not used for classification.

2.6.3 BUPA.

The goal of this data set is to predict the amount a patient drinks given certain measurements of their liver health. While this classification task may seem counter-intuitive, it is certainly harder than predicting a patient's liver health knowing how much they drink [11]. There are 345 instances, 5 numerical features, and 1 selector field which is not used for classification.

2.6.4 CRX.

The goal of this data set is to predict whether an applicant will be approved for a loan given some information on their finances [33]. There are 690 observations, 6 numerical features, and 9 categorical features.

2.6.5 Glass.

The goal of this data set is to predict whether a specimen of glass came from a window (either car or building window) or if it is non-window glass (containers, lamps, etc) by examining its chemical makeup [7]. There are 214 observations, 9 numerical features, and 1 ID field which is not used for classification.

2.6.6 Haberman.

The goal of this data set is to predict whether a patient will survive five years or longer after an operation given some very basic information about them- age, year of operation, number of positive axillary nodes detected. Because so little information is available about each patient, determining their survival chances is a difficult task [14]. There are 306 observations and 3 numerical features.

2.6.7 Iris.

This is the "classic" classification task, introduced by R.A. Fisher in 1936. The goal of this data set is to predict the sub-species of an iris given its petal length and width and its sepal length and width [9]. There are 150 observations and 4 numerical features.

2.6.8 Mammographic Masses.

The goal of this data set is to predict whether a mass detected in a mammography is benign or malignant given some measures from a computer aided diagnostic system [6]. There are 961 observations, 3 integer/ordinal features, and 2 categorical features.

2.6.9 Parkinson's.

The goal of this data set is to predict whether a patient has Parkinson's disease given some measurements of their voice [25]. There are 197 observations and 23 numerical features.

2.6.10 Pima Indians Diabetes.

The goal of this data set is to predict if a patient has diabetes given some measurements that are related to diabetes [41]. The patient set is restricted to females who are members of Pima Indian heritage [41]. There are 768 observations and 8 numerical features.

2.6.11 Spambase.

The goal of this data set is to predict if an e-mail is spam given some characteristics of the e-mail such as word frequencies, special character frequencies, and continuous runs of capital letters [18] [18]. There are 4,601 observations and 57 numerical features.

2.6.12 SPECTF.

The goal of this data set is to diagnose heart patients with features extracted from Single Proton Emission Computed Tomography (SPECT) images [24]. There are 534 observations and 44 numerical features.

2.6.13 Transfusion.

The goal of this data set is to predict whether a person donated blood in March 2007 given some information regarding their previous blood donations [46]. There are 748 observations and 4 numerical features.

2.6.14 WDBC.

This data set is very similar to the Breast Cancer Wisconsin data set, it has the same goal (diagnosing tumors) and was created by the same people [42]. However, the measurements taken of the tumors are different [42]. There are 569 observations, 30 numerical features, and 1 sample ID number which is not used for classification.

III. Methodology

This chapter outlines our experimental procedure used to look for a relationship between diversity and ensemble performance. First we use Monte Carlo resampling to provide an empirical distribution of classifier performance for six classifier types on fourteen different data sets. We use the resampling results to capture the expected change in classifier performance between training and validation sets. Second we train the same six classifier types on the original (non-resampled) training data. We evaluate every ensemble of three classifiers at multiple classification thresholds and with five different fusion techniques. We propose a new method for re-scoring class probabilities that allows the use of different classification thresholds for each classifier in the ensemble, a technique which was not previously possible with current techniques. Multiple diversity metrics for every ensemble and threshold combination are collected, as well as ensemble performance measures. Finally we analyze the experimental results for a relationship between the diversity metrics and ensemble performance.

To begin with we will look at the change in ensemble accuracy between test and validation sets to see if it compares to the changes we saw with individual classifiers in the bootstrapping results. We will look at the correlation between diversity measures and ensemble accuracy. We will create regression models to see the relative effects of ensemble test accuracy and ensemble test diversity on predicting ensemble validation accuracy, the maximization of which is the ultimate goal of classification. We will also how some simple ensemble selection schemes, such as selecting ensembles based solely on accuracy and selecting based solely on diversity, perform against an "oracle."

3.1 Data sets

The fourteen data sets described in section 2.5 are used in these experiments. On data sets that were already split between training and test sets, the data was aggregated into one large set. After there were fourteen monolithic training sets each set was centered and scaled to have a mean of zero and a standard deviation of one. The data sets were then each split into a training set (40%), a test set (30%), and a validation set (30%). The proportions were selected to keep the test and validation sets large enough that a few "difficult" exemplars would not have too large of an adverse effect on the accuracy. The data sets used were large enough that 40% was still large enough to provide an adequate size training set.

3.2 Classifiers

The six classifiers discussed in section 2.2.1 are used in these experiments. Four of the classifiers were implemented in R [34] using various packages(LDA/QDA[43], MLP[43], k-NN[43], and SVM[5]), and two of the classifiers were implemented with MATLAB's Neural Network Toolbox [28] (RBFN, PNN) because they did not have readily available implementations in R.

3.3 Diversity Measures

There were six different diversity measures used in these experiments- correlation, Yule's Q statistic, disagreement, double fault, entropy, and Kohavi-Wolpert Variance. These measures were selected to allow for comparison with prior research, these are the most used measures in published research so they provide for the largest amount of cross comparison.

3.4 Fusion methods

There were six different fusion methods used to create ensembles in these experiments- majority vote, mean fusion (BEM), weighted mean fusion (GEM), product rule, minimum rule, and maximum rule. These methods were selected because they appear frequently in published research so they provide for large amounts of cross comparison; they also have efficient implementations which is desirable when working with the amount of data used in this thesis.

3.5 Monte Carlo Resampling

Before evaluating an ensemble of classifiers, the individual classifiers themselves must be evaluated. To do this, we randomly partitioned the data into training, test, and validation sets thirty different times. The data was split 40/30/30% in the training, test, and validation sets. We trained the classifiers on the thirty training sets. We then created predictions for the corresponding test and validation sets and compared these predictions to the ground truth of the test and validation sets. This allowed us to create an empirical distribution of the accuracy of each classifier for each data set and also test for any difference between the test and validation sets. We did not expect any difference between test and validation sets because they are selected randomly, however, we wanted to ensure that our data partitioning and classification algorithms did not have any unusual behavior. Knowing the distributions of individual classifier accuracy can also allow us to perform a sanity check with our results in the main experiment. If our classifier accuracy in the main experiment was much different than the accuracies seen in this Monte Carlo experiment then we would need to investigate the cause. The code used to perform this bootstrapping experiment is attached in appendix B for the R code and appendix C for the MATLAB code.

3.6 Ensemble Combinations

The primary goal of this research is to discover if a relationship between ensemble accuracy and diversity exists, and to do that there must be a number of ensembles to evaluate. One area not examined in prior research is how diversity and accuracy relate at different classification thresholds. Most prior research only looked at a single classification threshold (0.5) and when they did vary the thresholds it was only presented as ROC curves of single classifier accuracy and MCS accuracy; no prior research has looked at how diversity changes with varying thresholds. The ensembles we construct not only vary the classification threshold, but also vary the classification threshold independently for each classifier by using our proposed alternate scoring technique.

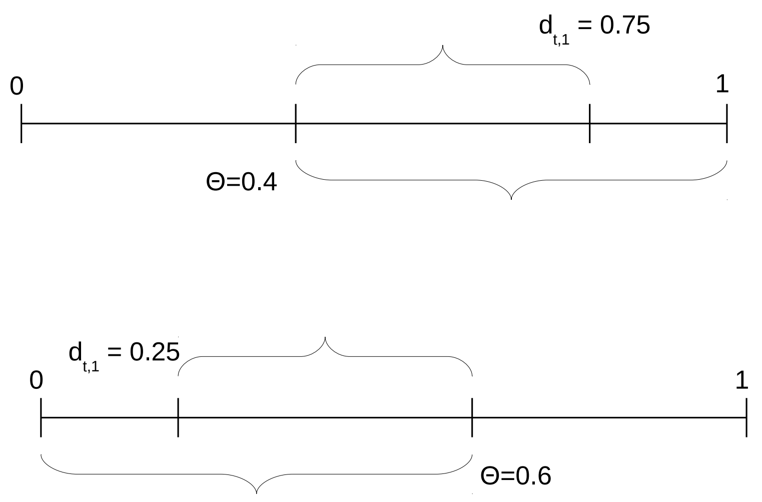
3.6.1 Alternate Scoring Technique.

The proposed alternate scoring technique transforms class probabilities into new scores by selection of a classification threshold, θ. The procedure takes classifier t's output probability of an observation belonging to class 1 $d_{t,1}$, and re-scores it to $d^*_{t,0}$ and $d^*_{t,1}$. The score not only captures the predicted class for an exemplar but also the relative distance of the original classifier score to the selected classification threshold:

$$d^*_{t,0} = max(0, \frac{\theta - d_{t,1}}{\theta})$$
$$d^*_{t,1} = max(0, \frac{d_{t,1} - \theta}{1 - \theta})$$

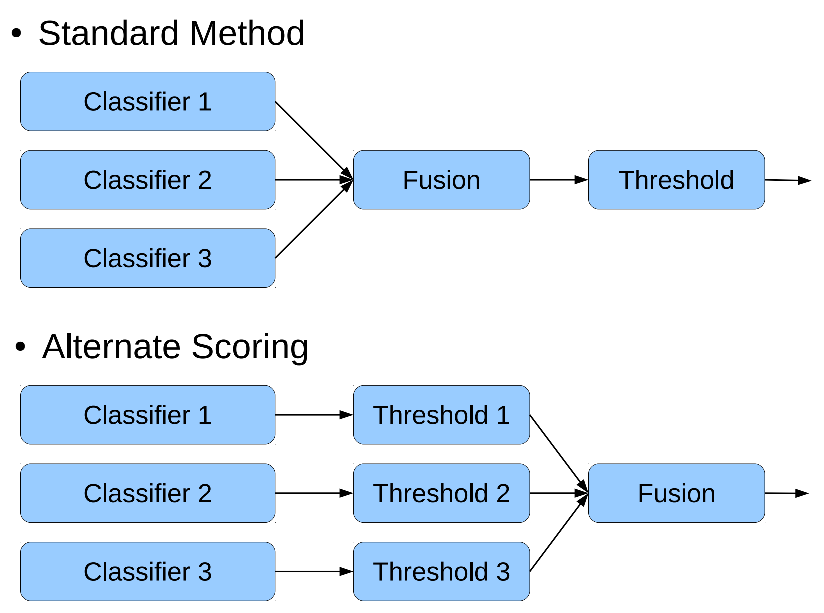
For an individual classifier, an assignment to class 0 would occur if $d^*_{t,0} > d^*_{t,1}$, and an assignment to class 1 would occur if $d^*_{t,0} \leq d^*_{t,1}$. A pictorial view of two examples is shown in figure 6, once where $d_{t,1} > \theta$, and once where $d_{t,1} < \theta$. The alternate scoring technique will be applied to the classifier outputs prior to performing fusion,

Figure 6. Graphical Representation of Alternate Scoring Technique



as opposed to the standard method which compares thresholds after performing fusion. The procedural flow of the two methods is compared in 7. These new scores do not behave like class probabilities because they do not sum to 1, but they are restricted to the interval [0,1]. Because the scores all fall on the same interval, we can perform the same fusion techniques on them as we could on class probabilities. We expect the performance of creating ensembles using this alternate scoring technique to perform similarly to ensembles created using class probabilities. For at least one case, fusing with mean fusion and all $\theta = 0.5$, we know ensembles created with the alternate scoring technique to perform exactly the same as ensembles created with class probabilities. A proof of this is provided in appendix A. A graphical comparison of the benefits of the alternate scoring technique is shown in figures 8 and 9. Figures 8 and 9 come from the same ensemble. A single threshold exists as a single point on the surface, and the range of thresholds provided by the standard method exists as a diagonal line. The alternate scoring technique can reach the entire surface

Figure 7. Graphical Comparison of Standard Method and Alternate Scoring



allowing more in-depth exploration of diversity. Being able to reach new areas of the classification space is the reason for using this alternate scoring technique, being able to explore a greater space may provide more insight into the relationship between accuracy and diversity.

3.6.2 Creating ensembles.

To evaluate the relationship between the accuracy and diversity of an ensemble, a number of ensembles must be produced. Different ensembles can be produced by including different classifiers in the ensemble or by varying the classification thresholds of the classifiers in the ensemble. Figure 10 shows the creation process for one ensemble. A function was created that takes the test and validation class probabilities from three classifiers, three individual classification thresholds as well as the ground truth and performs the alternate scoring technique, calculates the diversity metrics,

Figure 8. Sample accuracy surface over a range of thresholds

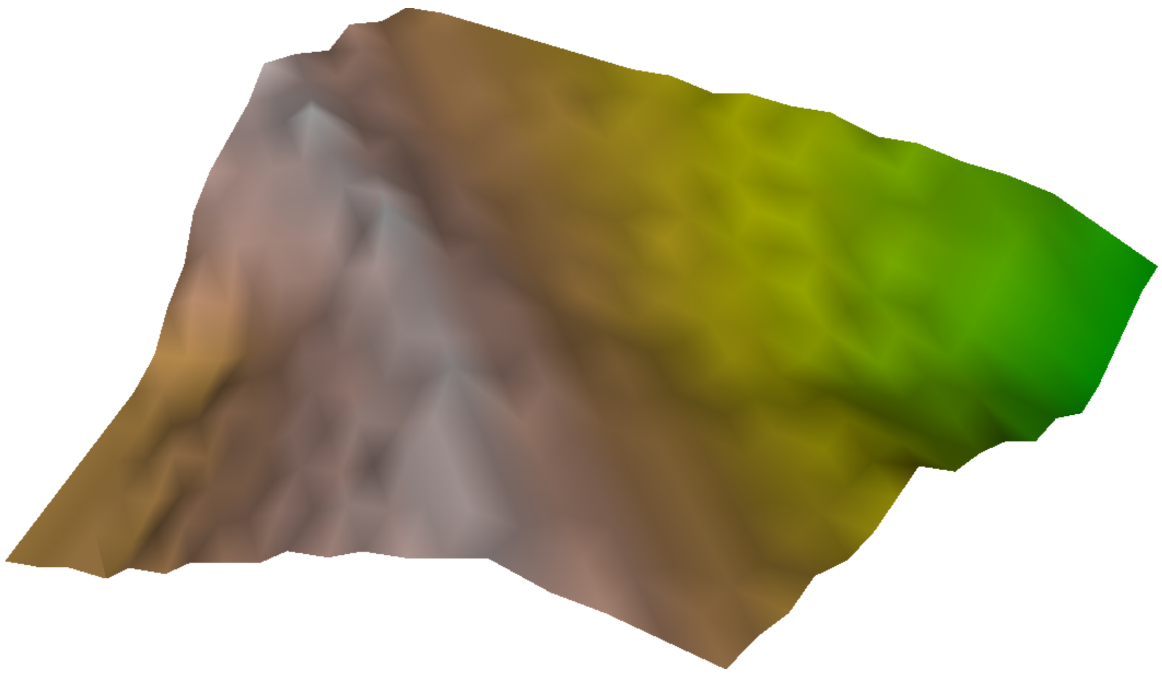


Figure 9. Sample diversity surface over a range of thresholds

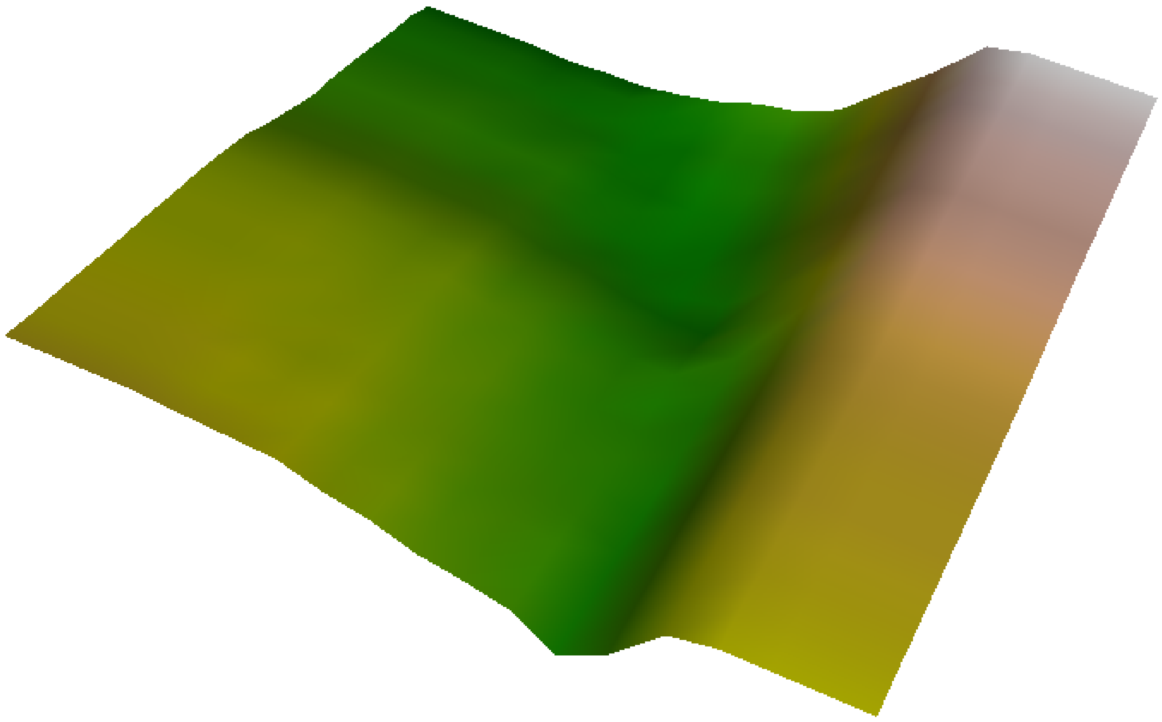
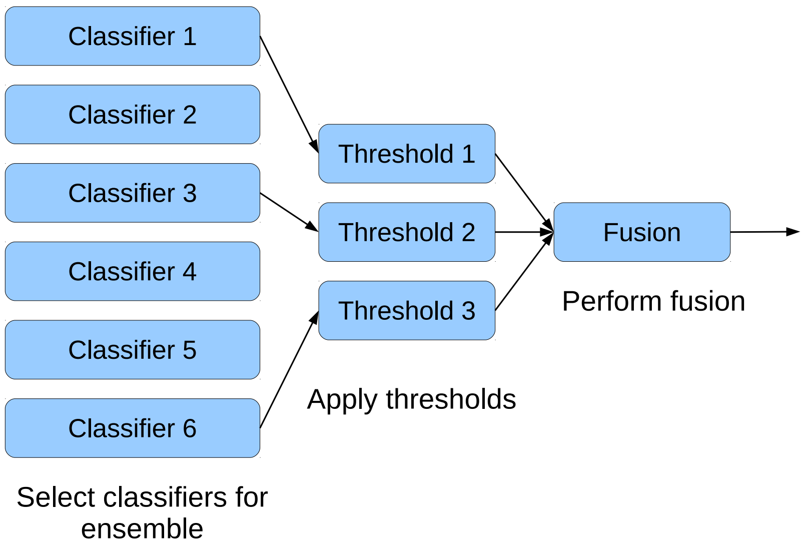


Figure 10. Visual Representation of Ensemble Creation



performs the fusion techniques, and also returns the performance metrics of the fused ensembles. This function can be thought of as a "wrapper" that takes the required inputs of an ensemble and returns the desired performance metrics; TCA, TP, FP; and the desired diversity metrics; Correlation, Yule's Q, Disagreement, Double Fault, Entropy, and Kohavi-Wolpert Variance.

$$f(test, validation, classifier\ 1, classifier\ 2, classifier\ 3, \theta_1, \theta_2, \theta_3, truth) = TCA, TP, FP, \rho, Q, D, DF, E, KW$$

Every possible ensemble with 3 classifiers was evaluated at every threshold from 0.05 to 0.95 with threshold step sizes of 0.05. The returned diversity metrics and ensemble performances were saved in a list for analysis. The code used to create all of the ensemble combinations is attached in appendix B for the R code and appendix C for the MATLAB code.

3.6.3 Computation complexity.

A note should be made about the computational complexity of this experiment. While it is simple to discuss creating *every* possible ensemble combination, the actual time it takes to perform this experiment is significant. The size of ensembles included is a combinatorics issue, and the complexity is exponential with respect to the size of the ensembles. With six classifiers, there are twenty ensembles of three different classifiers, and with three individual classification thresholds there are $19^3 = 6859$ different thresholds combinations for each ensemble. With fourteen data sets, there are 1,920,520 unique ensembles created. Each ensemble has six different diversity measures calculated and six different fusion techniques performed. Each diversity measure and fusion technique require a number of calculations and logical comparisons (comparisons are known to be very slow operations) that are related to the number of observations in the test and validation sets. Because of the large number of calculations the actual calculations were performed "in the cloud" using the Amazon Web Services Elastic Compute Cloud (AWS EC2). An AWS EC2 cluster instance enables the calculations to be finished in approximately 6 hours where a desktop workstation would have taken approximately 5 days of continuous computation to complete the experiment.

3.6.4 Comparison to ensembles created without the scoring technique.

Along with creating all ensembles with the alternative scoring technique, all ensembles were created without using the scoring technique. As mentioned above, using the alternative scoring technique gives identical performance when all classification thresholds are 0.5, but we hope to show that the alternative scoring technique compares favorably with the standard method at other classification thresholds as well.

3.7 Looking for Relationships

There are a number of different ways to look for a relationship between accuracy and diversity after creating all ensemble combinations and recording the diversity metrics and ensemble accuracies. One step that was taken for all procedures was to map the diversity metrics to the interval [0,1] where 0 is minimum diversity and 1 is maximum diversity. This is the same interval that accuracies fall on so their relative effects can be compared directly. Some diversity metrics already meet this criteria, such as disagreement and entropy. The rest of the diversity metrics are mapped in the following manner:

$$\begin{aligned} \rho^* &= 1 - |\rho| \\ Q^* &= 1 - |Q| \\ DF^* &= 1 - df \\ KW^* &= 3 \times KWV \end{aligned}$$

3.7.1 Correlations.

The first logical step to uncovering a relationship between diversity and accuracy is to determine if there is a linear correlation between the diversity metrics collected and the ensemble accuracies. The correlation between test set diversity and test set accuracies should be examined to see if there is a within set correlation, and the correlation between test set diversity and validation set accuracies should be examined to see if there exists any between set correlation.

3.7.2 Regression.

Another possible way to uncover a relationship between diversity and accuracy is to perform a linear regression on the results. If there is a relation between diversity and accuracy then the validation set accuracy may be able to be predicted by test set diversity (which would be very useful in ensemble building). It is probable that test set accuracy is the main predictor of validation set accuracy and that diversity may only explain some of the residual error. To determine if this is the case four regressions will be performed on each data set- one with diversity as the only regressor, one with accuracy as the only regressor, one with both diversity and accuracy as regressors, and one with diversity and accuracy as regressors including their interaction. The regression results will be examined to determine the effect of test set diversity and accuracy on validation accuracy. To account for the effects of which diversity metric is being used, what data set the ensemble came from, and and which fusion technique was used on the ensemble dummy variables are encoded. These dummy variables are included as as main effects to allow for a change in the regression intercept, and also interacted with accuracy and diversity to allow for the coefficients for accuracy and diversity to change based on the ensemble's properties (diversity metric, data set, and fusion technique). A regression of validation set accuracy with test set accuracy and diversity as the regressors would look like this without dummy variables:

$$TC\hat{A}_{validation} = \beta_0 + \beta_1 TCA_{test} + \beta_2 Diversity$$

This regression does not take into account the diversity metric, data set, and fusion technique in use. The full regression with dummy variables would look like this:

$$\begin{aligned} TC\hat{A}_{validation} = \beta_0 &+ \beta_1 TCA_{test} + \beta_2 Diversity + \beta_3 D_1 + \beta_4 D_2 + \beta_5 D_3 + \beta_{13} TCA_{test} D_1 \\ &+ \beta 14 TCA_{test} D_2 + \beta 15 TCA_{test} D_3 + \beta 23 Diversity D_1 + \beta 24 Diversity D_2 \\ &+ \beta 25 Diversity D_3 \end{aligned}$$

Where D_1 is the vector of dummy variables associated with which diversity metric is used, D_2 is the vector of dummy variables associated with the data set the ensemble comes from, and D_3 is the vector of dummy variables associated with the fusion technique used. The β_j's and $\beta_i j$'s associated with dummy variables are a vector as well. This full regression with dummy variables not only allows for the change of intercept and coefficients depending on the dummy variables and their interactions, it allows for testing the statistical significance of the dummy variables and the information they are associated with. For instance, if all of the coefficients associated with D_3, the fusion technique dummy variables, were insignificant, then we could conclude that the relationship between test set accuracy and validation set accuracy does not depend on the fusion technique. Likewise, if the coefficients associated with D_1, the diversity metric dummy variables, were insignificant, then we could conclude that there is no relationship between diversity and accuracy of an MCS.

3.7.3 Ensemble selection.

To examine the utility of diversity to determine classifiers membership in an ensemble three ensemble selection schemes will be evaluated and compared against the "oracle," which knows the best possible ensemble and threshold combination. The oracle is a realistic option in this experiment because we have evaluated every possible ensemble combination, so we know with 100% certainty which ensemble is the best. The first scheme will select the ensemble that has the highest ensemble test accuracy. The second scheme will select the ensemble that has the three classifiers with the highest individual test accuracy. The third scheme will select the ensemble with the highest test diversity. These schemes will be performed with each fusion type and their validation set accuracy will be compared to the best ensemble's validation accuracy as determined by the oracle. These comparisons will be placed in percentages for relative comparison across fusion techniques, diversity measures, and data sets. If diversity is a useful metric to select classifiers for an ensemble then the selection schemes that use diversity should compare favorably against the selection schemes that use accuracy.

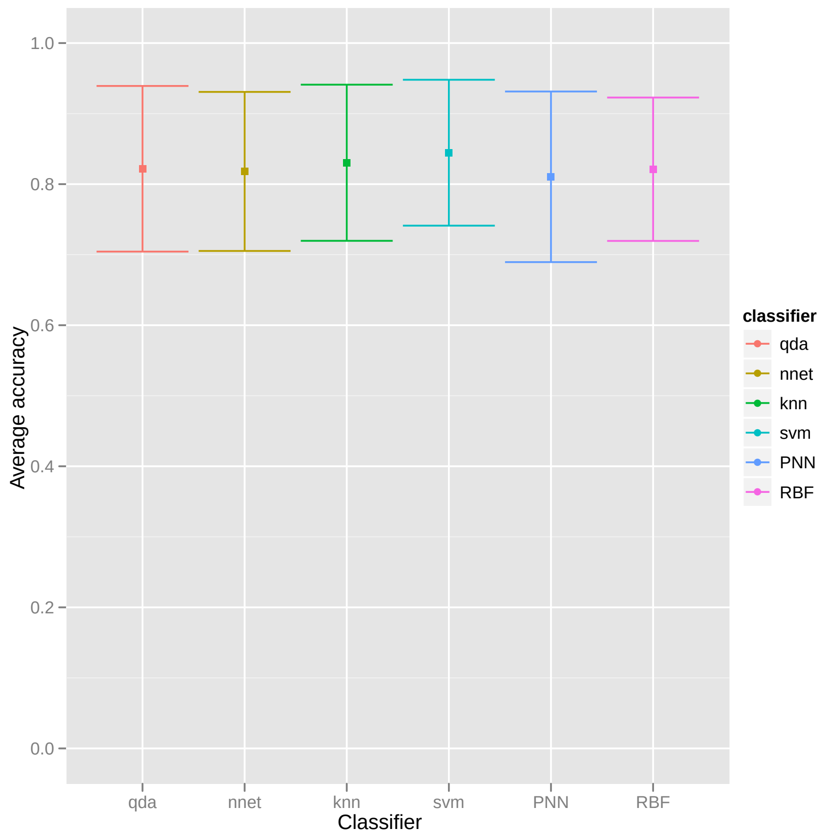
IV. Results and Analysis

4.1 Monte Carlo Resampling

The Monte Carlo Resampling results came out as expected, which was that all classifiers were capable of achieving over 50% TCA and there were no statistical differences between test and validation sets. Each classifier type managed an acceptable level of accuracy, each classifier's average accuracy is shown in figure 11. In all following plots, the error bars show

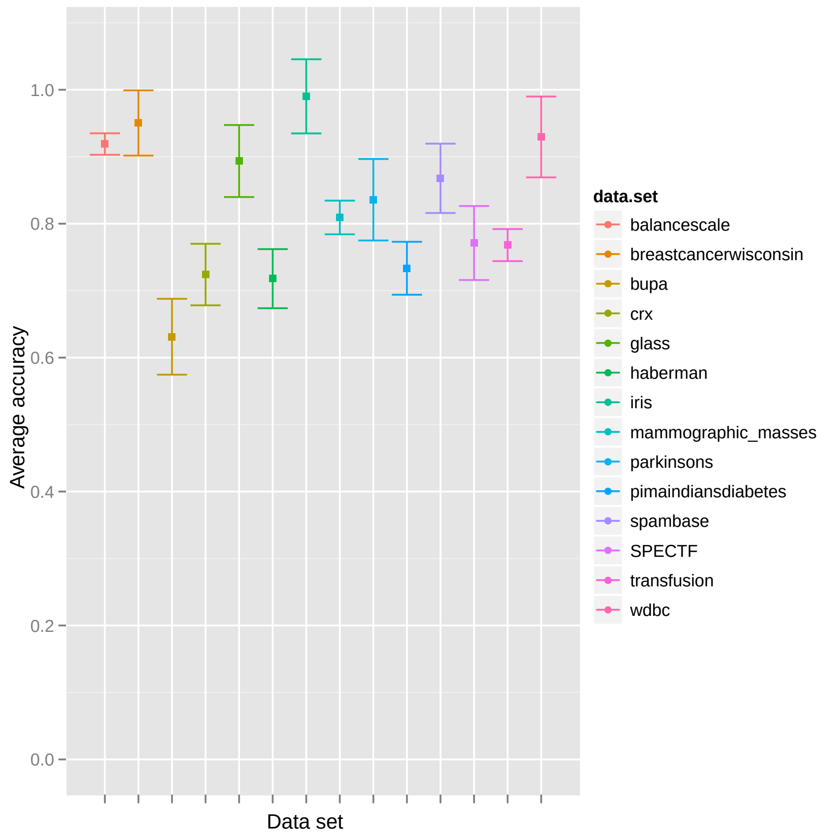
$pm1$ standard deviation from the mean. This is each classifier type's accuracy averaged across all data sets. The average accuracy of each data set was also acceptable,

Figure 11. Accuracy by Classifier



indicating that there were not any data sets that were "too hard" for classification. The average accuracy of each data set is shown in figure 12. This is the classification accuracy for each data set, averaged across all classifier types. On average, there

Figure 12. Accuracy by Data set



is no statistical difference between the test set accuracy and validation set accuracy for individual classifiers with a classification threshold of 0.5. This is shown in figure 13. Similarly, there is no statistical difference between the test set accuracy and validation set accuracy for each data set. The accuracy difference of each data set is shown in figure 14.

Figure 13. Difference between test and validation set by Classifier

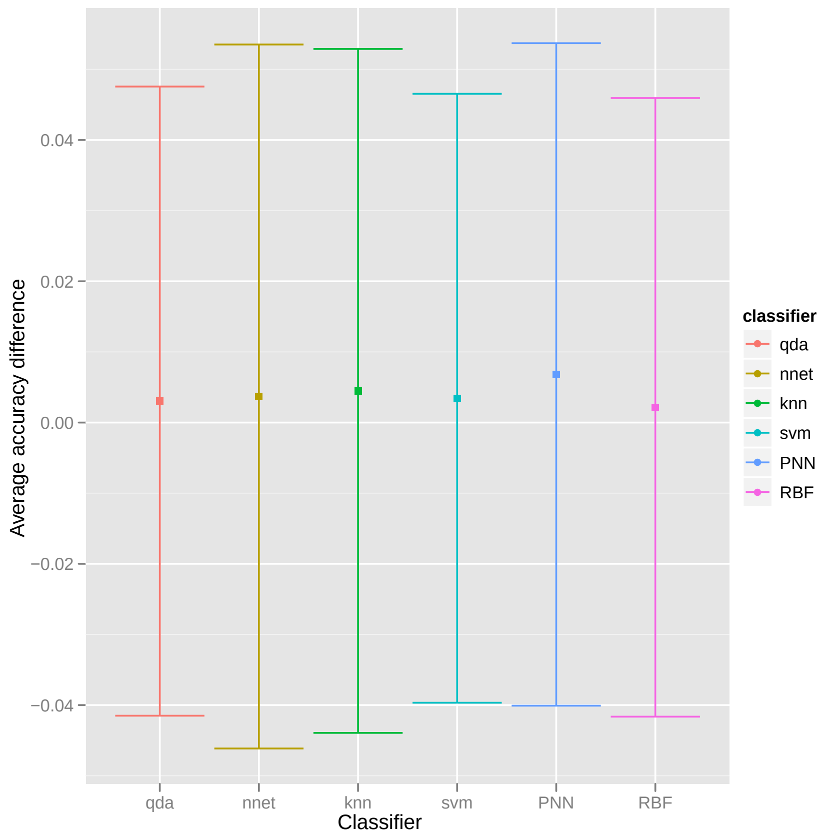
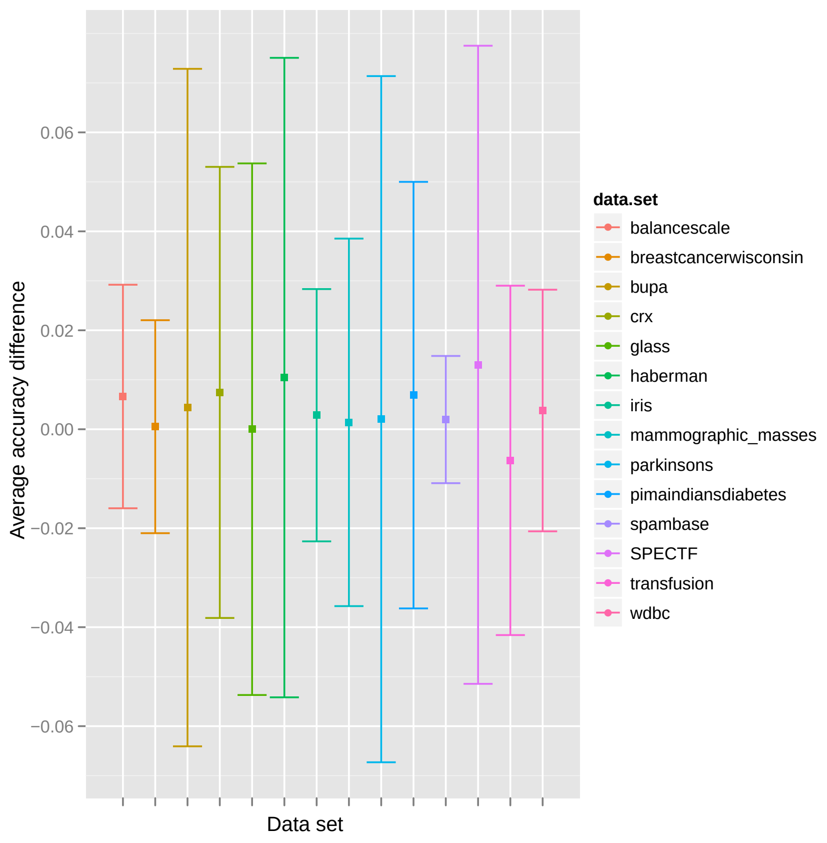


Figure 14. Difference between test and validation set by Data set



4.2 Alternative scoring technique

We believe that the alternate scoring technique performed adequately. For three of the fusion techniques; BEM, GEM, and Product Rule; the alternate scoring technique was able to achieve a higher level of accuracy. The two remaining fusion techniques, MIN and MAX, the alternate fusion technique did not achieve a very high level of accuracy. This is likely due to how the alternate scoring technique forces one of the scores to become zero which can greatly affect the behavior of these statistics. Table 7 shows a comparison of the alternate scoring technique's maximum and average performance separated by which fusion technique was applied, but averaged across all data sets. It is apparent that the alternate scoring technique has the potential

Table 7. Comparison of standard method to alternative scoring technique- achieved accuracy

Fusion	Max- standard	Max- alt	Avg- standard	Avg- alt
BEM	0.871	0.874	0.811	0.762
GEM	0.867	0.872	0.802	0.755
PRO	0.864	0.867	0.750	0.738
MIN	0.864	0.637	0.750	0.574
MAX	0.869	0.579	0.808	0.474

to perform as well as the standard method but loses some accuracy in the "tails" as the accuracy of the alternate scoring technique averaged across the range of classification thresholds is lower than the standard method. The actual performance of alternate scoring technique is not of great importance, the primary reason for applying this technique is to allow us to examine a greater range of classification threshold combinations and a greater range of diversity.

4.2.1 Diversity increase.

Using the alternate scoring technique allowed us to achieve a greater range of diversity, it was hoped that this greater range of diversity achieved would allow us to gain greater insight into the relationship between the accuracy and diversity of an MCS. Table 8 shows that the alternate scoring technique achieves a higher range of diversity for every diversity metric. The diversity ranges are averaged across all data sets in Table 8. This averaging across data sets is not an issue because there was never a data set that showed a decrease in diversity from using the alternate scoring technique.

Table 8. Comparison of standard method to alternative scoring technique- achieved diversity range

Metric	Range- Standard	Range- Alternate
Correlation	0.9243274	0.9673894
Yule's Q	0.9550785	0.9824356
Double-Fault	0.4078787	0.4240594
Disagreement	0.5488286	0.6529416
Entropy	0.8232429	0.9794124
K-W variance	0.5488286	0.6529416

4.3 Ensemble Combinations

The results of creating every ensemble combination using the alternative scoring technique are presented below.

4.3.1 Correlations.

Perhaps the simplest way to discover a relationship between ensemble and diversity accuracy is to calculate the correlation coefficient between them. For each diversity metric and fusion method we calculated the Pearson's r coefficient between the test diversity and test accuracy to see if there was any within set correlation. The Pearson's r coefficient between the test diversity and validation accuracy was also examined to see if there was any between set correlation that could possibly be exploited for ensemble selection. The correlation aggregated by diversity metric is perhaps the most informative, and is shown in figure 9. The correlation for

Table 9. Correlations by Diversity Metric

Diversity Metric	Within Set Correlation	Between Sets Correlation
Corr	0.023	-0.035
DF	0.352	0.238
Disag	-0.106	-0.124
Ent	-0.106	-0.124
KWV	-0.106	-0.124
Yule	0.001	-0.042

all diversity metrics is small, and for most of the metrics the sign is opposite what the conventional wisdom states. The conventional wisdom says that higher diversity should lead to higher accuracy and therefore have a positive correlation, but most of the correlation coefficients here are negative. A possible explanation is offered by Kuncheva in [21], where she shows how for most of the accuracy range diversity does have a negative correlation with accuracy, but once accuracy gets above a certain (fairly high) threshold, the relationship turns around and becomes a positive correlation. This relationship is shown in figure 15. Since we examined the relationship over the entire diversity range, we see mostly negative correlation and not much of

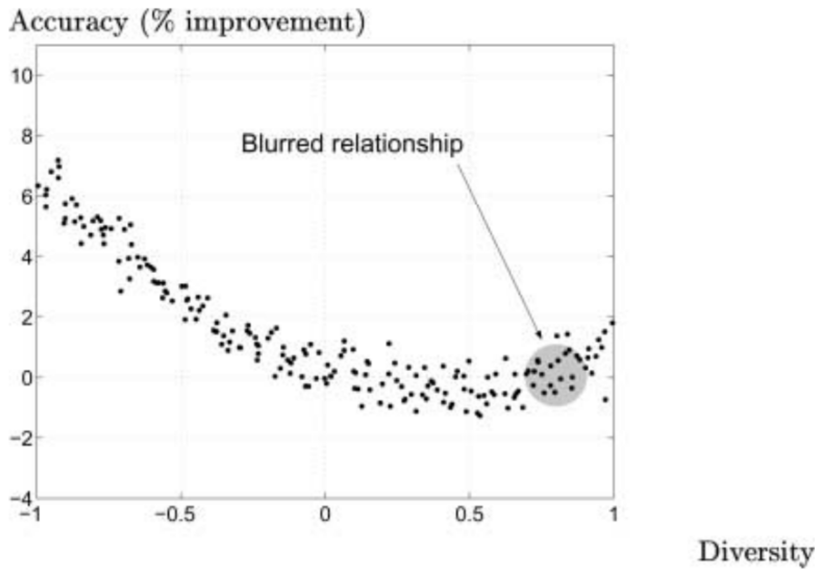


Figure 15. Reprinted from Kuncheva [21]

the tiny region where the relationship is positive. However, if the threshold where the correlation changes from negative to positive is not known we cannot exploit any correlation between diversity and accuracy. The one encouraging result from this is the positive and (relatively) high correlation between the double fault metric and accuracy. However, this correlation may be spurious because the double fault metric can be thought of as an indirect measure of ensemble accuracy as it is a measure of diversity since it only takes into account wrong classifications; there is an inherent link between incorrect classifications and accuracy.

4.3.2 Regression results.

If there is a relationship between diversity and accuracy, how much effect does diversity have on accuracy anyway? That question might be answered by performing a linear regression with ensemble accuracy as the response. Since we are most interested in ensemble creation, we use ensemble validation set accuracy as the response and measures from the test set as the regressors. This allows us to emulate a real world application of picking an ensemble based on our test set performance and treating the validation set as new observations that are classified after an ensemble is selected. We performed three regressions- using test set accuracy as the only regressor, using test set diversity as the only regressor, and using both test set accuracy and diversity as regressors. In each regression, dummy variables for the diversity metric were coded and interacted to allow for a change of intercept and to allow each coefficient to change for the different diversity metrics. All of the dummy variables associated with the diversity metric were significant, indicating that there is a statistically significant relationship between the diversity and accuracy of an MCS. Dummy variables for the fusion technique and data set were also included; while they were also significant, they were not interpreted. The primary interest was the coefficients for accuracy and diversity. The results of the regressions are presented below in table 10, including the "interesting" coefficients as well as two measures of prediction performance, the coefficient of determination (R^2) and root mean square error (RMSE). Readily ap-

Table 10. Regression Coefficients + Results

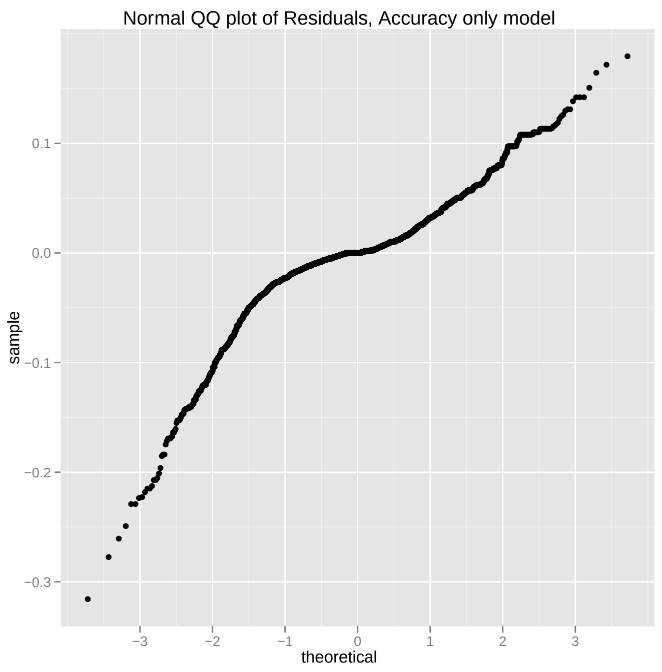
Model	Accuracy	Corr	Yule	DF	Disag	Ent	KWV	R^2	RMSE
Accuracy Only	0.987	N/A	N/A	N/A	N/A	N/A	N/A	0.932	0.0404
Diversity Only	N/A	0.0425	0.0451	0.223	-0.0264	0.0126	-0.0264	0.729	0.0841
Accuracy + Diversity	0.983	-0.00503	0.0022	-0.00759	-0.00425	-0.00039	-0.00425	0.933	0.0402

parent is that the model that uses diversity as the only regressor has the lowest R^2 and highest RMSE. The models that include accuracy as a regressor have much lower error, but it seems that including diversity as a regressor adds very little value when accuracy is already included in the model. Since all of the regressors are bounded on the same interval [0,1], their coefficients can be directly compared to look at the effect of accuracy and each diversity metric. It is apparent that test set accuracy has a far greater effect on the validation set accuracy than any of the diversity metrics, indicating that even a large change in test diversity can only affect a small change in validation set accuracy.

4.3.2.1 Model adequacy.

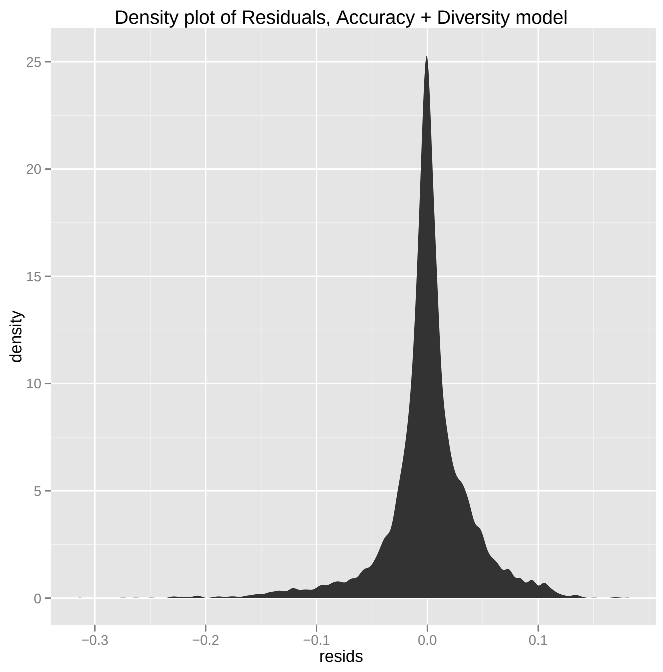
A large amount of data was included in these regressions, but no amount of data can make a validate a regression model; it still must pass model adequacy checks. The most important assumptions that must be met is that the errors are normally distributed, with a mean of zero and a constant variance. Checking to see if the residuals are normally distributed is usually done by plotting the residuals on a normal quantile plot and applying the "fat pencil test." The normal quantile plot of the regression with accuracy as the sole regressor is shown in figure 16. All of the normal quantile plots for the each of the three regressions looked similar to figure 16, indicating that the residuals are not normally distributed, but they could be called normally distributed except with thin tails. Another way of looking at the normalcy

Figure 16. Normal probability plot, Accuracy only model



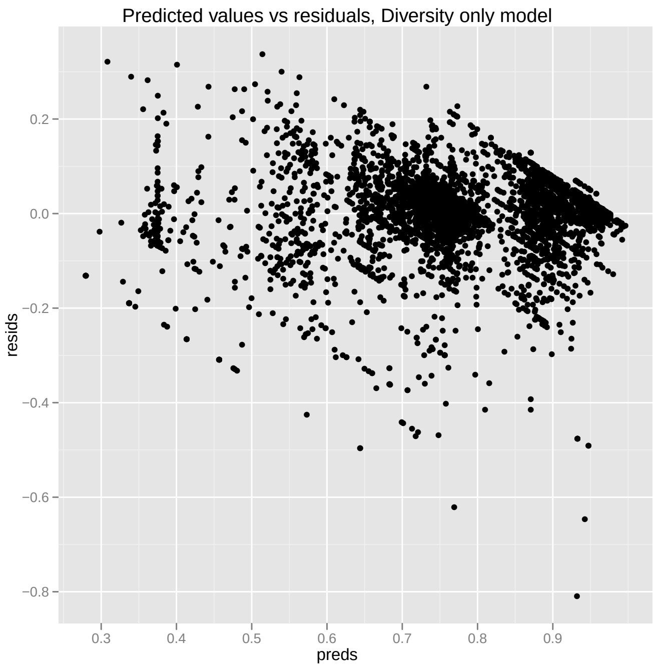
residuals is with a density plot, as shown in figure 17. This is the density plot of the residuals from the regression model that included both accuracy and diversity. As with the normal quantile plots, all three regression residuals looked similar to this. The density plot also shows that the residuals appear to be centered at zero. The

Figure 17. Residual Density plot, Accuracy + Diversity model



other main assumption that must be met is that the residuals have constant variance. The primary method of diagnosing this is to look at a scatter plot of the residuals by predicted values. A plot of the residuals by predicted values is shown in figure 18. Again, all three of the predicted by residual plots looked similar to this. The predicted by residual plots did not show any non-constant variance. It may look like there is a "cone" in the plot, but it is important to note that the central area of the plot is a very dense cloud of thousands of points, and the points on the outside that make up the cone shape is a relatively Least squares regression is known to be robust as long as the residuals are "mound shaped" and do not need to be ex-

Figure 18. Predicted values vs residuals, Diversity only model



actly normally distributed. The residuals easily meet this criteria for being "mound shaped." Another assumption, that the regressors are uncorrelated, was not checked because there is no effective measure for checking multicollinearity on a model with interaction terms, but multicollinearity was checked on initial models that did not include interaction terms and multicollinearity was not an issue then.

4.3.3 Ensemble Selection results.

Because we created every possible ensemble combination we knew exactly which one of the ensembles was optimal for classifying the validation set. We selected ensembles based on measurements from the test set and compared the performance those ensembles achieved to the best ensemble selected by the "oracle." For example, if the selection criteria was "ensemble accuracy," for each fusion method the ensemble with the best test set ensemble accuracy was selected. Each selected ensemble's validation accuracy was compared to the best possible validation accuracy achievable by that fusion method, and the percent performance was recorded. The selection criteria used were ensemble test accuracy, individual classifier accuracy, and all six test diversity metrics. The percent performance that each selection criteria achieved, aggregated by fusion method, is shown in figure 11. As shown, selecting ensembles

Table 11. Selection Performance by Fusion Technique

	Accuracy		Diversity					
Fusion Technique	Ensemble	Individual	Corr	Yule	DF	Disag	Ent	KWV
BEM	94%	95%	90%	90%	93%	91%	91%	91%
GEM	93%	93%	87%	89%	91%	86%	86%	86%
MAX	95%	95%	90%	88%	91%	91%	91%	91%
MIN	95%	93%	78%	80%	78%	66%	66%	66%
MVOTE	93%	95%	86%	86%	93%	83%	83%	83%
PROD	95%	93%	78%	80%	78%	66%	66%	66%

based on accuracy gives the highest performance, while selecting ensembles based on diversity gives lower performance. As a rule of thumb, the double fault diversity metric performed the best out of all the diversity metrics. This analysis shows that test set accuracy should be the primary criteria for selecting ensembles, either the ensemble accuracy or the accuracies of the individual classifiers is adequate. If there are two ensembles that tie in accuracy criteria, diversity may be useful as a secondary criteria to break the tie.

V. Conclusions

5.1 Research Contribution

1. Examined the relationship between diversity and accuracy in classification regions not previously explored by researchers.

2. Confirmed that much of the prior knowledge about diversity and accuracy holds true in these previously unexplored regions.

3. Shows that there may be some relationship between diversity and ensemble accuracy, but it is too small to exploit at ensemble creation time.

4. Proposes a new technique for classifier scoring that allows for multiple classification thresholds with fusion techniques that previously only allowed a single classification threshold. Provides proof that this technique performs similarly to single threshold techniques, so there is no drawback.

5.2 Limitations

The ensemble selection techniques discussed forced the ensembles to contain exactly three classifiers, and did not care about including more or fewer classifiers in the ensemble. It also did not look at including one classifier more than once with different classification thresholds. This could create a classifier fused with itself, and creating diversity with itself. This may or may not provide an increase to ensemble accuracy.

5.3 Conclusion

This study took an in depth look at the relationship between diversity and accuracy. An alternative scoring technique was proposed to create diversity but may also be useful for creating more accurate more robust ensembles. The new scoring technique merits further research on its own, its primary purpose in this paper was contributing towards diversity research. A statistically significant relationship between diversity and ensemble performance was shown through the results, however the contribution that diversity makes to accuracy is too small to be practically useful. It was shown that selecting ensembles based on diversity alone does not perform as well as selecting ensembles based on accuracy alone, and the regression results show that diversity makes too small of a contribution to ensemble accuracy to make up for all but the smallest changes in individual classifier accuracy.

5.4 Further Research

1. Expand the research in this thesis to include more classifiers, diversity metrics, and fusion techniques.

2. Look at how artificial diversity creation techniques within individual classifiers (random seeds, bootstrapping, etc.) compare to diversity between different classifier types (SVM vs MLP, k-NN vs QDA, etc.).

3. Look for non-linear relationships between diversity and ensemble accuracy. Consider using multiple classification thresholds with the alternative scoring technique to target high diversity and accuracy.

4. Look at more in depth ensemble selection techniques that include ensembles of varying sizes, allow classifiers to appear in ensemble more than once, and allow for diversity to make up for small differences in accuracy.

5. Further examine alternative scoring technique to see if it can be used for tuning ensembles to increase accuracy.

Appendix A. Proof of Alternate Scoring Techniques Equivalence to the Standard Method Under Certain Conditions

1.1 Introduction

Presented below is a proof that mean fusion of the suggested alternate scoring method is equivalent to the Basic Ensemble Model (BEM) of the class probabilities under certain conditions. The proof here is restricted to two classifiers and a two-class problem. Generalizing the proof to more than two classifiers, more than two classes, or other fusion techniques is left to the reader.

1.2 Symbols and Terminology

- Class Probability- The support given by a classifier to an observation for a certain class. Class probabilities must fall between 0 and 1, and the class probabilities for a single observation must sum to 1.

- Class Score- The support given by a classifier to an observation for a certain class, but is not constrained to fall between 0 and 1. Greater class scores are interpreted as higher support. The alternative class scoring method abstracts class probabilities into class scores.

- Class 1- The "positive" class. In a two class problem, class 1 is the class that is supported. Generally, this class is interpreted as being the more important of the two classes.

- Class 0- The "negative" class. The other class of a two class problem. Generally, this class is interpreted as being the more mundane of the two classes.

- P_i- The class probability assigned by the ith classifier to Class 1 for an observation. P_{BEM} is the class probability obtained after performing BEM on the ensemble of classifiers.

- Q_i- The class probability assigned by the ith classifier to Class 0 for an observation. By definition, $Q_i = 1 - P_i$.

- R_i- The class score assigned by the ith classifier to Class 1 for an observation.

- S_i- The class score assigned by the ith classifier to Class 0 for an observation. There is not a fixed relationship between R_i and S_i as there is between P_i and Q_i.

- θ_i- The threshold selected for scoring observations from classifier i. θ_{BEM} is the threshold selected for classifying observations from a BEM ensemble.

1.3 Basic Ensemble Model

For detailed information on BEM, see [31]. Essentially, BEM is the mean of the class probabilities given by each classifier for a particular class. For example, P_{BEM} for an ensemble of two classifiers would be performed as follows:

$$P_{BEM} = \frac{P_1 + P_2}{2}$$

Q_{BEM} can be calculated in a similar way, but it may be simpler to just calculate $Q_{BEM} = 1 - P_{BEM}$. BEM creates an ensemble with less error than either individual classifier and is a simple but effective method of fusing the outputs of multiple classifiers [31]. Final classification is performed by selecting a threshold, θ_{BEM}, and if $P_{BEM} \geq \theta_{BEM}$ then the observation is labeled as Class 1. If $P_{BEM} < \theta_{BEM}$, then the observation is labeled as Class 0. Note that this technique only uses one threshold, which is applied *after* BEM is performed. Also, note that this method preserves the relationship $Q_{BEM} = 1 - P_{BEM}$.

1.4 Alternative Scoring Method

The alternative scoring method proposed by Butler and Friend (this thesis research) abstracts the class probabilities given by a classifier into class scores. First, a threshold is selected, θ_i, for each classifier. The scoring method creates a score for each class from the relative distance of P_i from θ_i in the following manner:

$$R_i = max(0, \frac{P_i - \theta_i}{1 - \theta_i})$$
$$S_i = max(0, \frac{\theta_i - P_i}{\theta_i})$$

Note that with this scoring method one of the class scores will always be 0, with the opposing class score being positive (unless $P_i = \theta_i$ in which case both scores will be 0). Mean fusion on the class scores is performed similarly to BEM:

$$R_{mean} = \frac{R_1 + R_2}{2}$$
$$S_{mean} = \frac{S_1 + S_2}{2}$$

Classification is performed by comparing R_{mean} and S_{mean}. If $R_{mean} \geq S_{mean}$ then the observation is labeled as Class 1. If $R_{mean} < S_{mean}$ then the observation is labeled as Class 0. Note that with this method multiple thresholds are selected during the scoring method, allowing for more flexibility when combining classifiers. However, the added flexibility may not be useful if it cannot perform at least as well as existing fusion methods. Mean fusion of the scores from the alternative scoring method can perform at least as well as BEM, since they are equivalent when $\theta_{BEM} = \theta_1 = \theta_2 = \theta$. The following is a proof.

1.5 Proof

The following section proves that BEM and mean fusion of the scores from the alternative scoring method are equivalent when $\theta_{BEM} = \theta_1 = \theta_2 = \theta = 0.5$. This is a proof of two classifier ensembles only. The key to the proof is looking at how each classification is made. With BEM, Class 1 is assigned when $P_{BEM} \geq \theta_{BEM}$ (Class 0 is assigned otherwise); with mean fusion of the scores from the alternative scoring method, Class 1 is assigned when $R_{mean} \geq S_{mean}$ (Class 0 is assigned otherwise). To prove the two methods are equivalent, we must prove that the comparison of P_{BEM} to θ_{BEM} is equivalent to the comparison of R_{mean} to S_{mean}. There are three separate cases:

1. Both P_i's$\geq \theta$
2. One of the P_i's$\geq \theta$ and one of the P_i's$< \theta$
3. Both P_i's$< \theta$

1.5.1 Case 1.

Case 1 is trivial. When P_1 and P_2 are both greater than or equal to θ, then

$$P_{BEM} = \frac{P_1 + P_2}{2} \geq \theta_{BEM}$$

This means that BEM would label an observation falling into Case 1 as Class 1. Also, both R_1 and R_2 will be positive, while both S_1 and S_2 will be 0. Therefore, R_{mean} will be positive and S_{mean} will be 0, which means that

$$R_{mean} \geq S_{mean}$$

This means that the mean fusion of the scores from the alternative scoring method would also label an observation falling into Case 1 as Class 1. Therefore, the two methods are equivalent under Case 1.

1.5.2 Case 2.

Case 2 is more involved, and should be broken down into two sub-cases. Each case has common ground in that one of the P_i's is $\geq \theta$ and one is $< \theta$. While the greater P_i could be P_1 or P_2, let $P_1 \geq \theta$ and $P_2 < \theta$ for the purposes of this proof. Simply swap P_1 and P_2 and create two new sub-cases that are symmetrical to theses two sub-cases for completeness. The two sub cases are:

1. $P_1 \geq \theta$ and $P_2 < \theta$ while having $P_1 - \theta \geq \theta - P_2$
2. $P_1 \geq \theta$ and $P_2 < \theta$ while having $P_1 - \theta < \theta - P_2$

1.5.2.1 Case 2- Sub-Case 1.

For $P_1 \geq \theta$ and $P_2 < \theta$ while having $P_1 - \theta \geq \theta - P_2$ we can show that $P_{BEM} \geq \theta$ since the distance between P_1 and θ is larger than the distance between P_2 and θ.

Starting with inequality (1) and performing some algebra we can show:

$$\begin{aligned} P_1 - \theta &\geq \theta - P_2 \qquad (1) \\ \frac{P_1}{2} - \frac{\theta}{2} &\geq -\left(\frac{P_2}{2} - \frac{\theta}{2}\right) \\ \left(\frac{P_1}{2} - \frac{\theta}{2}\right) + \left(\frac{P_2}{2} - \frac{\theta}{2}\right) &\geq 0 \\ \frac{P_1 + P_2}{2} &\geq \theta \\ P_{BEM} &\geq \theta \end{aligned}$$

Since $P_{BEM} \geq \theta$, we would label an observation falling into Case 2 Sub-Case 1 as Class 1 using BEM. Also, knowing that $P_1 \geq \theta$ means that $R_1 \geq 0$ and $S_1 = 0$. Similarly, knowing that $P_2 < \theta$ means that $R_2 = 0$ and $S_2 \geq 0$. Then $R_{mean} = \frac{R_1}{2}$ and $S_{mean} = \frac{S_2}{2}$. Substituting the formulas in for R_1 and S_2 yields:

$$\begin{aligned} R_{mean} &= \frac{P_1 - \theta}{2\theta} \\ S_{mean} &= \frac{\theta - P_2}{2\theta} \end{aligned}$$

By definition of Sub-Case 1 $P_1 - \theta \geq \theta - P_2$, therefore $R_{mean} \geq S_{mean}$, which would label an observation falling into Case 2 Sub-Case 1 as Class 1 using mean fusion of the scores from the alternative scoring method. Since both methods would label Case 2 Sub-Case 1 as Class 1, the two methods are equivalent under Case 2 Sub-Case 1.

1.5.2.2 Case 2- Sub-Case 2.

For $P_1 \geq \theta$ and $P_2 < \theta$ while having $P_1 - \theta < \theta - P_2$ we can show that $P_{BEM} < \theta$ because the distance between P_1 and θ is smaller than the distance between P_2 and

θ. Starting with inequality (2) and performing some algebra we can show:

$$\begin{aligned} P_1 - \theta &< \theta - P_2 \qquad (2) \\ \frac{P_1}{2} - \frac{\theta}{2} &< -\left(\frac{P_2}{2} - \frac{\theta}{2}\right) \\ \left(\frac{P_1}{2} - \frac{\theta}{2}\right) + \left(\frac{P_2}{2} - \frac{\theta}{2}\right) &< 0 \\ \frac{P_1 + P_2}{2} &< \theta \\ P_{BEM} &< \theta \end{aligned}$$

Since $P_{BEM} < \theta$ we would label an observation falling into Case 2 Sub-Case 2 as Class 0 using BEM. Also, knowing that $P_1 \geq \theta$ means that $R_1 \geq 0$ and $S_1 = 0$. Similarly, knowing that $P_2 < \theta$ means that $R_2 = 0$ and $S_2 \geq 0$. Then $R_{mean} = \frac{R_1}{2}$ and $S_{mean} = \frac{S_2}{2}$. Substituting the formulas in for R_1 and S_2 yields:

$$R_{mean} = \frac{P_1 - \theta}{2\theta}$$
$$S_{mean} = \frac{\theta - P_2}{2\theta}$$

By definition of Sub-Case 2 $P_1 - \theta < \theta - P_2$, therefore $R_{mean} < S_{mean}$, which would label an observation falling into Case 2 Sub-Case 2 as Class 0 using mean fusion of the scores from the alternative scoring method. Since both methods would label Case 2 Sub-Case 1 as Class 0, the two methods are equivalent under Case 2 Sub-Case 2.

1.5.2.3 Case 2- Both sub-cases.

Since the two methods are equivalent under both sub-cases of Case 2, then the two methods are equivalent under Case 2.

1.5.3 Case 3.

Case 3 is the "complement" of Case 1 and is just as trivial. When P_1 and P_2 are both less than θ, then

$$P_{BEM} = \frac{P_1 + P_2}{2} < \theta_{BEM}.$$

This means that BEM would label an observation falling into Case 3 as Class 0. Also, both R_1 and R_2 will be 0, while both S_1 and S_2 will be positive. Therefore, R_{mean} will be 0 and S_{mean} will be positive, meaning

$$R_{mean} < S_{mean}.$$

This means that the mean fusion of the scores from the alternative scoring method would also label an observation falling into Case 3 as Class 0. This proves the two methods are equivalent under Case 3.

1.6 All three cases

Since the alternate scoring technique is equivalent to the standard method under all three possible cases, the alternate scoring technique is equivalent to the standard method. This means they will output the same class labels and have the same classification accuracy, true positive rate, false positive rate, and the same values for all diversity measures.

1.7 Threshold restriction

Since both methods are equivalent under all three cases, then it has been proven that they are equivalent methods– but only when all the thresholds selected are equal tp 0.5: $\theta_{BEM} = \theta_1 = \theta_2 = 0.5$. If the thresholds are not all equal to 0.5, it is readily apparent that the two methods are not equivalent.

1.8 Conclusion

This paper provided a proof that BEM and mean fusion of the scores from the alternative scoring method are equivalent when creating an ensemble of two classifiers for a two class problem; also, the thresholds selected need to be equal. This is advantageous because this means that the alternative scoring method will always be able to produce classifications that are at least as accurate as BEM, while allowing for more choice in the classification parameters in order to produce classifications that are more accurate than BEM on occasion. While this proof was only for ensembles of two classifiers and two class problems, the reader should be able to generalize the proof to include more classifiers, more classes, or other fusion techniques.

Appendix B. R code

All of the R code is listed here. There are three main sections, the code for the Monte Carlo resampling experiment, the code for the main ensemble creation experiment, and the universal functions used by both experiments. In the Monte Carlo resampling code, there are frequent references to "bootstrap," this is a carry over from previous work, what is really meant is "Monte Carlo," but changing the names causes a bunch of errors. Also, only the source code from one data set (parkinsons) is shown. This is representative of all 14 data sets, the only difference in the files is the file name and which columns of the data set are selected as the input/exemplar matrix and which column is selected as the response/truth vector.

Listing B.1. "R code for Monte Carlo resampling experiment- parkinsonsBootstrap.R"

```
############### header type stuff ###################

#import required packages
library(caret)
library(class)
library(e1071)
library(MASS)
library(nnet)

#set working directory to where the files are
setwd('~/ThesisData')

#load functions from source code
source('thesisfunctions.R')
```

```
############### this part here changes for each dataset
    #############

file = 'parkinsons'
#read in data- file name varies per dataset.
data = read.csv(file=paste('~/ThesisData/CSVfiles/',file,'.
    csv', sep=''), header=F)

#split off data and response- columns vary per dataset.
Y = data[,18]
data[,18] = NULL
data[,1] = NULL
X = data

# is dataset rank deficient across classes? if so, set
    ldaflag
ldaflag = TRUE

#Turn class labels into binary vector. R treats factors with
     a
#base-1 ideology, we will just turn that into a numeric
    vector and
#subtract off 1, to make it a vector of 0s and 1s.  This is
    up in the
#specific part of the code because some datasets have already
     done
```

```
#this in the .csv file and this line needs to be commented
    out.
Y = as.numeric(Y)

################ stuff after here does not change
    #################

bootnum = 30 #number of times to bootstrap

source('ResampleBootstrap.R')
source('MatlabExportBootstrap.R')
source('MatlabImportBootstrap.R')
source('TrainBootstrap.R')
source('PredictBootstrap.R')
source('DistBootstrap.R')



##head back up to top directory
setwd('~/ThesisData')

##save image
save(file=paste(file,'Bootstrap.RData', sep=''), list=ls())
```

Listing B.2. "R code for Monte Carlo resampling experiment- ResampleBootstrap.R"

```
#Turn X into model matrix and center and scale data BEFORE
    resampling
X = model.matrix(~.-1, X)
```

```
X = scale(X, center=T, scale=T)

#Split data into training, validation, and test sets.  First
    split
#off test set (30%), then from the remaining data, split off
    3/7 to
#become the validation set, and the remaining is the training
     set.
#This results in a split of train/val/test = 40/30/30 %
for(i in 1:bootnum){
#create data split 40/30/30%
trainvalind =createDataPartition(Y, times=1, p=0.7)
eval(parse(text=paste('Ytest', i,' = Y[-trainvalind$Resample1
    ]', sep='')))
eval(parse(text=paste('Xtest', i,' = X[-trainvalind$Resample1
    ,]', sep='')))
eval(parse(text=paste('Ytrainval', i,' = Y[trainvalind$
    Resample1]', sep='')))
eval(parse(text=paste('Xtrainval', i,' = X[trainvalind$
    Resample1,]', sep='')))
eval(parse(text=paste('valind = createDataPartition(Ytrainval
    ', i,', times=1, p=0.4286)', sep='')))#3/7=0.4286
eval(parse(text=paste('Ytrain', i,' = Ytrainval', i,'[-valind
    $Resample1]', sep='')))
eval(parse(text=paste('Xtrain', i,' = Xtrainval', i,'[-valind
    $Resample1,]', sep='')))
eval(parse(text=paste('Yval', i,' = Ytrainval', i,'[valind$
    Resample1]', sep='')))
eval(parse(text=paste('Xval', i,' = Xtrainval', i,'[valind$
    Resample1,]', sep='')))

#do row names to keep SVM happy
eval(parse(text=paste('row.names(Xtrain', i,') = 1:length(row
    .names(Xtrain', i,'))', sep='')))
eval(parse(text=paste('row.names(Xtest', i,') = 1:length(row.
    names(Xtest', i,'))', sep='')))
eval(parse(text=paste('row.names(Xval', i,') = 1:length(row.
    names(Xval', i,'))', sep='')))

}
```

Listing B.3. "R code for Monte Carlo resampling experiment- MatlabExportBootstrap.R"

```
library(R.matlab)

setwd('~/ThesisData/MATLABinputfiles')
#write training stuff to .mat file for MATLAB to use
commandstring = paste('writeMat(\'', file, 'Bootstrap.mat\'',
     sep='')
for(i in 1:bootnum){
commandstring = paste(commandstring, ', Xtrain', i, '=Xtrain'
    , i, ', Ytrain', i, '=Ytrain', i, sep='')
commandstring = paste(commandstring, ', Xtest', i, '=Xtest',
    i, ', Xval', i, '=Xval', i, sep='')
}
commandstring = paste(commandstring, ')', sep='')

eval(parse(text=commandstring))

#send system command to run Matlab code
system(paste('~/matlab -nosplash -nodesktop -r \"filename = \
    '', file , '\', cd ~/ThesisData/, NNetBootstrap , quit\"',
    sep=''))

setwd('~/ThesisData')
```

Listing B.4. "R code for Monte Carlo resampling experiment- MatlabImportBootstrap.R"

```
library(R.matlab)

setwd('~/ThesisData/MATLABoutputfiles')

#import .mat file data into R
MATLAB = readMat(paste(file , 'Bootstrap.mat', sep=''))

#read MATLAB data into individual variables
for(i in 1:bootnum){
commandstring = paste('PNNtestpreds',i,' = MATLAB$PNN',i,'
    testout', sep='')
eval(parse(text=commandstring))
commandstring = paste('RBFtestpreds',i,'␣=␣MATLAB$RBF',i,'
    testout', sep='')
eval(parse(text=commandstring))
commandstring = paste('PNNvalpreds',i,'␣=␣MATLAB$PNN',i,'
    valout', sep='')
eval(parse(text=commandstring))
commandstring = paste('RBFvalpreds',i,'␣=␣MATLAB$RBF',i,'
    valout', sep='')
eval(parse(text=commandstring))
}

setwd('~/ThesisData')
```

Listing B.5. "R code for Monte Carlo resampling experiment- TrainBootstrap.R"

```
##########train classifiers on training samples
    ################

for(i in 1:bootnum){
#Quadratic/Linear Discriminant Analysis
#if dataset is rank deficient, use lda- else use qda
commandstring = paste('if(ldaflag){qdamod',i,'␣=␣lda(Xtrain',
    i,',␣Ytrain',i,')}␣else␣{qdamod',i,'␣=␣qda(Xtrain',i,',␣
    Ytrain',i,')}',sep='')
eval(parse(text=commandstring))

#Feed Foward Neural Net- MLP, 1 hidden layer of size 3
commandstring = paste('nnetmod',i,' = nnet(Xtrain',i,', cbind
    (Ytrain',i,', 1-Ytrain',i,'), size=3)',sep='')
eval(parse(text=commandstring))

#k-NN requires no training

#SVM- linear kernel, default options from e1071
commandstring = paste('svmmod',i,' = svm(Xtrain',i,', y=
    Ytrain',i,', scale = F, type=\'C-classification\', kernel
    =\'linear\', cost=1, probability=T)',sep='')
eval(parse(text=commandstring))
#PNN and RBF networks are done in MATLAB, I pick this back up
    in
#MATLAB import
}
```

Listing B.6. "R code for Monte Carlo resampling experiment- PredictBootstrap.R"

```
#######create predictions -- posterior probs############

for(i in 1:bootnum){

#QDA/LDA
commandstring = paste('qdatestpreds',i,' = predict(qdamod',i,
    ', Xtest',i,')$posterior[,2]', sep='')
eval(parse(text=commandstring))
commandstring = paste('qdavalpreds',i,' = predict(qdamod',i,'
    , Xval',i,')$posterior[,2]', sep='')
```

```
eval(parse(text=commandstring))

#FFNN
commandstring = paste('nnettestpreds',i,' = predict(nnetmod',
    i,', Xtest',i,', type=\'raw\')[,1]', sep='')
eval(parse(text=commandstring))
commandstring = paste('nnetvalpreds',i,' = predict(nnetmod',i
    ,', Xval',i,', type=\'raw\')[,1]', sep='')
eval(parse(text=commandstring))

#k-NN
commandstring = paste('knntestresults',i,' = knn(Xtrain',i,',
    Xtest',i,', Ytrain',i,', k=10, l=1, prob=T, use.all=F)',
    sep='')
eval(parse(text=commandstring))
commandstring = paste('knntestpreds',i,' = probs(
    knntestresults',i,')', sep='')
eval(parse(text=commandstring))

commandstring = paste('knnvalresults',i,' = knn(Xtrain',i,', 
    Xval',i,', Ytrain',i,', k=10, l=1, prob=T, use.all=F)',
    sep='')
eval(parse(text=commandstring))
commandstring = paste('knnvalpreds',i,' = probs(knnvalresults
    ',i,')', sep='')
eval(parse(text=commandstring))
```

```
#svm
commandstring = paste('svmtestpreds',i,'=attr(predict(svmmod'
    ,i,', Xtest',i,', probability=T), \'probabilities\')', sep
    ='')
eval(parse(text=commandstring))

commandstring = paste('svmvalpreds',i,'=attr(predict(svmmod',
    i,', Xval',i,', probability=T), \'probabilities\')', sep='
    ')
eval(parse(text=commandstring))

commandstring = paste('if(colnames(svmtestpreds',i,')[1]==\'
    1\'){svmtestpreds',i,' = svmtestpreds',i,'[,1]} else{
    svmtestpreds',i,' = svmtestpreds',i,'[,2]}', sep='')
eval(parse(text=commandstring))

commandstring = paste('if(colnames(svmvalpreds',i,')[1]==\'1\
    '){svmvalpreds',i,' = svmvalpreds',i,'[,1]} else{
    svmvalpreds',i,' = svmvalpreds',i,'[,2]}', sep='')
eval(parse(text=commandstring))
}
```

Listing B.7. "R code for Monte Carlo resampling experiment- DistBootstrap.R"

```
qdatestaccuracy = NULL
qdavalaccuracy = NULL
nnettestaccuracy = NULL
```

```
nnetvalaccuracy = NULL
knntestaccuracy = NULL
knnvalaccuracy = NULL
svmtestaccuracy = NULL
svmvalaccuracy = NULL
PNNtestaccuracy = NULL
PNNvalaccuracy = NULL
RBFtestaccuracy = NULL
RBFvalaccuracy = NULL


for(i in 1:bootnum){
commandstring = paste('qdatestaccuracy␣=␣rbind(
    qdatestaccuracy,␣accuracy(label(qdatestpreds',i,',␣0.5),␣
    Ytest',i,'))', sep='')
eval(parse(text=commandstring))

commandstring = paste('qdavalaccuracy␣=␣rbind(qdavalaccuracy,
    ␣accuracy(label(qdavalpreds',i,',␣0.5),␣Yval',i,'))', sep=
    '')
eval(parse(text=commandstring))

commandstring = paste('nnettestaccuracy␣=␣rbind(
    nnettestaccuracy,␣accuracy(label(nnettestpreds',i,',␣0.5),
    ␣Ytest',i,'))', sep='')
eval(parse(text=commandstring))
```

```
commandstring = paste('nnetvalaccuracy␣=␣rbind(
    nnetvalaccuracy,␣accuracy(label(nnetvalpreds',i,',␣0.5),␣
    Yval',i,'))', sep='')
eval(parse(text=commandstring))

commandstring = paste('knntestaccuracy␣=␣rbind(
    knntestaccuracy,␣accuracy(label(knntestpreds',i,',␣0.5),␣
    Ytest',i,'))', sep='')
eval(parse(text=commandstring))

commandstring = paste('knnvalaccuracy␣=␣rbind(knnvalaccuracy,
    ␣accuracy(label(knnvalpreds',i,',␣0.5),␣Yval',i,'))', sep=
    '')
eval(parse(text=commandstring))

commandstring = paste('svmtestaccuracy␣=␣rbind(
    svmtestaccuracy,␣accuracy(label(svmtestpreds',i,',␣0.5),␣
    Ytest',i,'))', sep='')
eval(parse(text=commandstring))

commandstring = paste('svmvalaccuracy␣=␣rbind(svmvalaccuracy,
    ␣accuracy(label(svmvalpreds',i,',␣0.5),␣Yval',i,'))', sep=
    '')
eval(parse(text=commandstring))
```

```
commandstring = paste('PNNtestaccuracy␣=␣rbind(
    PNNtestaccuracy,␣accuracy(label(PNNtestpreds',i,',␣0.5),␣
    Ytest',i,'))', sep='')
eval(parse(text=commandstring))

commandstring = paste('PNNvalaccuracy␣=␣rbind(PNNvalaccuracy,
    ␣accuracy(label(PNNvalpreds',i,',␣0.5),␣Yval',i,'))', sep=
    '')
eval(parse(text=commandstring))

commandstring = paste('RBFtestaccuracy␣=␣rbind(
    RBFtestaccuracy,␣accuracy(label(RBFtestpreds',i,',␣0.5),␣
    Ytest',i,'))', sep='')
eval(parse(text=commandstring))

commandstring = paste('RBFvalaccuracy␣=␣rbind(RBFvalaccuracy,
    ␣accuracy(label(RBFvalpreds',i,',␣0.5),␣Yval',i,'))', sep=
    '')
eval(parse(text=commandstring))
}
```

Listing B.8. "R code for ensemble creation experiment- parkinsons.R"

```
############## header type stuff ##################

#import required packages
library(caret)
library(class)
library(e1071)
library(MASS)
library(nnet)

#set working directory to where the files are
setwd('~/ThesisData')

#load functions from source code
source('thesisfunctions.R')

############### this part here changes for each dataset
  ############

file = 'parkinsons'
#read in data- file name varies per dataset.
data = read.csv(file=paste('~/ThesisData/CSVfiles/',file,'.
  csv', sep=''), header=F)

#split off data and response- columns vary per dataset.
Y = data[,18]
data[,18] = NULL
data[,1] = NULL
X = data

# is dataset rank deficient across classes? if so, set
  ldaflag
```

```
ldaflag = TRUE

#Turn class labels into binary vector.  R treats factors with
    a
#base-1 ideology, we will just turn that into a numeric
   vector and
#subtract off 1, to make it a vector of 0s and 1s.  This is
   up in the
#specific part of the code because some datasets have already
    done
#this in the .csv file and this line needs to be commented
   out.
Y = as.numeric(Y)

################## stuff after here does not change
   ##################

source('Resample.R')
source('MatlabExport.R')
source('Train.R')
source('Predict.R')
source('MatlabImport.R')
source('Combine.R')
source('mcTripleTheta.R') #Parallelized version of original
   code for running
# on many-core cluster
```

```
##head back up to top directory
setwd('~/ThesisData')


##save image
save(file=paste(file,'.RData', sep=''), list=ls())
```

Listing B.9. “R code for ensemble creation experiment- Resample.R”

```
#Split data into training, validation, and test sets.  First
   split
#off test set (30%), then from the remaining data, split off
   3/7 to
#become the validation set, and the remaining is the training
    set.
#This results in a split of train/val/test = 40/30/30 %
trainvalind = createDataPartition(Y,times=1, p=0.7)
Ytest = Y[-trainvalind$Resample1]
Xtest = X[-trainvalind$Resample1,]
Ytrainval = Y[trainvalind$Resample1]
Xtrainval = X[trainvalind$Resample1,]
valind = createDataPartition(Ytrainval, times=1, p=0.4286)#3/
   7=0.4286
Ytrain = Ytrainval[-valind$Resample1]
Xtrain = Xtrainval[-valind$Resample1,]
Yval = Ytrainval[valind$Resample1]
Xval = Xtrainval[valind$Resample1,]
```

```
#garbage collect- since R is wasteful with memory, we don't
  want to
#keep data around that we don't need.
#rm(X, Y, Ytrainval, Xtrainval, trainvalind, valind)

#turn into model matrix
Xtest = model.matrix(~.-1,Xtest)
Xval = model.matrix(~.-1,Xval)
Xtrain = model.matrix(~.-1,Xtrain)

Xtest = scale(Xtest, center=T, scale=T)
Xtrain = scale(Xtrain, center=T, scale=T)
Xval = scale(Xval, center=T, scale=T)

#change row names to keep the svm classifier happy
row.names(Xtrain) = 1:length(row.names(Xtrain))
row.names(Xtest) = 1:length(row.names(Xtest))
row.names(Xval) = 1:length(row.names(Xval))
```

Listing B.10. “R code for ensemble creation experiment- MatlabExport.R”

```
#write files to .csv files for importing into MATLAB
write.csv(Xtrain, file=paste('MATLABinputfiles/',file,'-
  Xtrain.csv',
   sep=''), row.names=F)
write.csv(Xtest, file=paste('MATLABinputfiles/',file,'-Xtest.
  csv',
    sep=''), row.names=F)
write.csv(Xval, file=paste('MATLABinputfiles/',file, '-Xval.
   csv',
    sep=''), row.names=F)
write.csv(Ytrain, file=paste('MATLABinputfiles/',file, '-
   Ytrain.csv',
    sep=''), row.names=F)

#send system command to run Matlab code
system(paste('~/matlab -nosplash -nodesktop -r \"filename = \
   '', file , '\', cd ~/ThesisData , MATLABstuff, quit\"', sep=
   ''))
```

Listing B.11. "R code for ensemble creation experiment- Train.R"

```
##########train classifiers on training samples
   ###############

#Quadratic/Linear Discriminant Analysis
#if dataset is rank deficient, use lda- else use qda
if(ldaflag){
qdamod = lda(Xtrain, Ytrain)
} else {
qdamod = qda(Xtrain, Ytrain)
}

#Feed Foward Neural Net- MLP, 1 hidden layer of size 3
nnetmod = nnet(Xtrain, cbind(Ytrain, 1-Ytrain), size=3)
```

```
#k-NN requires no training


#SVM- linear kernel, default options from e1071
svmmod = svm(Xtrain, y=Ytrain, scale = F,
     type='C-classification', kernel='linear', cost=1,
         probability=T)


#PNN and RBF networks are done in MATLAB, I pick this back up
     in
#MATLAB import
```

Listing B.12. "R code for ensemble creation experiment- Predict.R"

```
#######create predictions -- posterior probs############
#QDA/LDA
qdatestpreds = predict(qdamod, Xtest)$posterior[,2]
qdavalpreds = predict(qdamod, Xval)$posterior[,2]


#FFNN
nnettestpreds = predict(nnetmod, Xtest, type='raw')
nnetvalpreds = predict(nnetmod, Xval, type='raw')


#k-NN


knntestresults = knn(Xtrain, Xtest, Ytrain, k=10, l=1, prob=T
    ,
     use.all=F)
```

```
knntestpreds = probs(knntestresults)

knnvalresults = knn(Xtrain, Xval, Ytrain, k=10, l=1, prob=T,
    use.all=F)
knnvalpreds = probs(knnvalresults)

#svm

svmtestpreds = attr(predict(svmmod, newdata=Xtest,
   probability=T),
    'probabilities')
if(colnames(svmtestpreds)[1]=='1'){
svmtestpreds = svmtestpreds[,1]
}else{
svmtestpreds = svmtestpreds[,2]
}

svmvalpreds = attr(predict(svmmod, newdata=Xval, probability=
   T),
    'probabilities')
if(colnames(svmvalpreds)[1]=='1'){
svmvalpreds = svmvalpreds[,1]
}else{
svmvalpreds = svmvalpreds[,2]
}
```

```
head(attr(predict(svmmod, newdata=Xtest, probability=T),'
    probabilities'))
```

Listing B.13. "R code for ensemble creation experiment- MatlabImport.R"

```
#readline(prompt = 'waiting for matlab results, ENTER when
    done ')

# import results from MATLAB
     setwd('~/ThesisData/MATLABoutputfiles')
     pnntestpreds = read.csv(paste(file,'-PNNtestpreds.csv',
         sep=''), header=F)
     pnnvalpreds = read.csv(paste(file,'-PNNvalpreds.csv',sep=
         ''), header=F)
     rbftestpreds = read.csv(paste(file,'-RBFtestpreds.csv',
         sep=''), header=F)
     rbfvalpreds = read.csv(paste(file,'-RBFvalpreds.csv',sep=
         ''), header=F)
     setwd('~/ThesisData')
#end matlab import
```

Listing B.14. "R code for ensemble creation experiment- Combine.R"

```
# create a "grand prediction matrix" that includes the
    posterior
# probability predictions from all 5 classifiers- both test
    and
# validation sets
```

```
grandtestpreds = cbind(qdatestpreds, nnettestpreds[,1],
   knntestpreds,
     svmtestpreds, pnntestpreds, rbftestpreds)
grandvalpreds = cbind(qdavalpreds, nnetvalpreds[,1],
   knnvalpreds,
     svmvalpreds, pnnvalpreds, rbfvalpreds)
```

Listing B.15. "R code for ensemble creation experiment- mcTripleTheta.R"

```
source('thesisfunctions.R')
load('mclist.RData')
load('mclist2.RData')

results = mclapply(1:nrow(mclist), function(x) wrapper3(
   mclist[x,1], mclist[x,2], mclist[x,3], grandtestpreds[,
   mclist[x,1]], grandtestpreds[,mclist[x,2]], grandtestpreds
   [,mclist[x,3]], grandvalpreds[,mclist[x,1]], grandvalpreds
   [,mclist[x,2]], grandvalpreds[,mclist[x,3]], mclist[x,4],
   mclist[x,5], mclist[x,6], Ytest, Yval), mc.preschedule=
   TRUE, mc.set.seed=FALSE, mc.silent=TRUE, mc.cores=32, mc.
   cleanup=TRUE)

rownames(results) = NULL
colnames(results) = NULL

results = simplify2array(results)
results = t(results)
```

```
colnames(results) = c('c1num', 'c2num', 'c3num', 'c1thresh',
  'c2thresh', 'c3thresh', 'c1testacc', 'c2testacc', '
  c3testacc', 'c1valacc', 'c2valacc', 'c3valacc', 'c1testtp'
  , 'c2testtp', 'c3testtp', 'c1testfp', 'c2testfp', '
  c3testfp', 'c1valtp', 'c2valtp', 'c3valtp', 'c1valfp', '
  c2valfp', 'c3valfp', 'testcorr', 'testyule', 'testdf', '
  testdisag', 'testrmsd', 'testent', 'testKWV', 'valcorr', '
  valyule', 'valdf', 'valdisag', 'valrmsd', 'valent', '
  valKWV', 'MVOTEtestacc', 'MVOTEtesttp', 'MVOTEtestfp', '
  BEMtestacc', 'BEMtesttp', 'BEMtestfp', 'GEMtestacc', '
  GEMtesttp', 'GEMtestfp', 'PROtestacc', 'PROtesttp', '
  PROtestfp', 'MINtestacc', 'MINtesttp', 'MINtestfp', '
  MAXtestacc', 'MAXtesttp', 'MAXtestfp', 'MVOTEvalacc', '
  MVOTEvaltp', 'MVOTEvalfp', 'BEMvalacc', 'BEMvaltp', '
  BEMvalfp', 'GEMvalacc', 'GEMvaltp', 'GEMvalfp', 'PROvalacc
  ', 'PROvaltp', 'PROvalfp', 'MINvalacc', 'MINvaltp', '
  MINvalfp', 'MAXvalacc', 'MAXvaltp', 'MAXvalfp')
```

Listing B.16. "Universal R code used in both experiments- thesisfunctions.R"

```
#bring in other wrapper file
source('Wrappers.R')
#wrapper function for getting lots of results from one
  command:
wrapper = function(testpreds_i, testpreds_j, valpreds_i,
  valpreds_j, Ytest, Yval, thresh){
   testlabels_i = label(testpreds_i, thresh)
   testlabels_j = label(testpreds_j, thresh)
   vallabels_i = label(valpreds_i, thresh)
   vallabels_j = label(valpreds_j, thresh)
  #grab diversity metrics
   testcorr = correlation(testlabels_i, testlabels_j, Ytest)
   valcorr = correlation(vallabels_i, vallabels_j, Yval)
   testyule = yuleq(testlabels_i, testlabels_j, Ytest)
   valyule = yuleq(vallabels_i, vallabels_j, Yval)
   testdf = df(testlabels_i, testlabels_j, Ytest)
   valdf = df(vallabels_i, vallabels_j, Yval)
   testdisag = disagreement(testlabels_i, testlabels_j)
   valdisag = disagreement(vallabels_i, vallabels_j)
   testent = entropy(testlabels_i, testlabels_j, Ytest)
   valent = entropy(vallabels_i, vallabels_j, Yval)
   testKWV = KWV(testlabels_i, testlabels_j, Ytest)
   valKWV = KWV(vallabels_i, vallabels_j, Yval)

   #do fusion on test results:
   bemresulttestpreds = (testpreds_i + testpreds_j)/2
   bemresultvalpreds = (valpreds_i + valpreds_j)/2
   #TODO: code up GEM
   prodresulttestpreds = testpreds_i * testpreds_j
   prodresultvalpreds = valpreds_i * valpreds_j
   minresulttestpreds = pmin(testpreds_i, testpreds_j)
   minresultvalpreds = pmin(valpreds_i, valpreds_j)
   maxresulttestpreds = pmax(testpreds_i, testpreds_j)
   maxresultvalpreds = pmax(valpreds_i, valpreds_j)
```

```
#create fused labels
bemtestlabels = label(bemresulttestpreds, thresh)
bemvallabels = label(bemresultvalpreds, thresh)
gemtestlabels = label(bemresulttestpreds, thresh)
gemvallabels = label(bemresultvalpreds, thresh)
prodtestlabels = label(prodresulttestpreds, thresh)
prodvallabels = label(prodresultvalpreds, thresh)
mintestlabels = label(minresulttestpreds, thresh)
minvallabels = label(minresultvalpreds, thresh)
maxtestlabels = label(maxresulttestpreds, thresh)
maxvallabels = label(maxresultvalpreds, thresh)

#create fused accuracy
bemtestacc = accuracy(bemtestlabels, Ytest)
bemvalacc = accuracy(bemvallabels, Yval)
gemtestacc = accuracy(gemtestlabels, Ytest)
gemvalacc = accuracy(gemvallabels, Yval)
prodtestacc = accuracy(prodtestlabels, Ytest)
prodvalacc = accuracy(prodvallabels, Yval)
mintestacc = accuracy(mintestlabels, Ytest)
minvalacc = accuracy(minvallabels, Yval)
maxtestacc = accuracy(maxtestlabels, Ytest)
maxvalacc = accuracy(maxvallabels, Yval)

#create fused tp, fp, tn, fn
```

```
bemtesttp = tpr(bemtestlabels, Ytest)
bemtestfp = fpr(bemtestlabels, Ytest)
bemtesttn = tnr(bemtestlabels, Ytest)
bemtestfn = fnr(bemtestlabels, Ytest)
bemvaltp = tpr(bemvallabels, Yval)
bemvalfp = fpr(bemvallabels, Yval)
bemvaltn = tnr(bemvallabels, Yval)
bemvalfn = fnr(bemvallabels, Yval)
gemtesttp = tpr(gemtestlabels, Ytest)
gemtestfp = fpr(gemtestlabels, Ytest)
gemtesttn = tnr(gemtestlabels, Ytest)
gemtestfn = fnr(gemtestlabels, Ytest)
gemvaltp = tpr(gemvallabels, Yval)
gemvalfp = fpr(gemvallabels, Yval)
gemvaltn = tnr(gemvallabels, Yval)
gemvalfn = fnr(gemvallabels, Yval)
prodtesttp = tpr(prodtestlabels, Ytest)
prodtestfp = fpr(prodtestlabels, Ytest)
prodtesttn = tnr(prodtestlabels, Ytest)
prodtestfn = fnr(prodtestlabels, Ytest)
prodvaltp = tpr(prodvallabels, Yval)
prodvalfp = fpr(prodvallabels, Yval)
prodvaltn = tnr(prodvallabels, Yval)
prodvalfn = fnr(prodvallabels, Yval)
mintesttp = tpr(mintestlabels, Ytest)
mintestfp = fpr(mintestlabels, Ytest)
```

```
mintesttn = tnr(mintestlabels, Ytest)
mintestfn = fnr(mintestlabels, Ytest)
minvaltp = tpr(minvallabels, Yval)
minvalfp = fpr(minvallabels, Yval)
minvaltn = tnr(minvallabels, Yval)
minvalfn = fnr(minvallabels, Yval)
maxtesttp = tpr(maxtestlabels, Ytest)
maxtestfp = fpr(maxtestlabels, Ytest)
maxtesttn = tnr(maxtestlabels, Ytest)
maxtestfn = fnr(maxtestlabels, Ytest)
maxvaltp = tpr(maxvallabels, Yval)
maxvalfp = fpr(maxvallabels, Yval)
maxvaltn = tnr(maxvallabels, Yval)
maxvalfn = fnr(maxvallabels, Yval)

return(c(bemtestacc, gemtestacc, prodtestacc, mintestacc,
    maxtestacc, testcorr, testyule, testdf, testdisag,
   testent, testKWV, bemvalacc, gemvalacc,
  prodvalacc, minvalacc, maxvalacc, valcorr, valyule,
     valdf, valdisag, valent, valKWV, bemtesttp,
     bemtestfp, bemtesttn, bemtestfn, bemvaltp, bemvalfp,
  bemvaltn, bemvalfn, gemtesttp, gemtestfp, gemtesttn,
     gemtestfn, gemvaltp, gemvalfp, gemvaltn, gemvalfn,
     prodtesttp, prodtestfp, prodtesttn,
        prodtestfn, prodvaltp, prodvalfp, prodvaltn, prodvalfn,
            mintesttp, mintestfp, mintesttn, mintestfn,
           minvaltp, minvalfp, minvaltn, minvalfn,
        maxtesttp, maxtestfp, maxtesttn, maxtestfn, maxvaltp,
           maxvalfp, maxvaltn, maxvalfn))
}

#wrapper2 function for ranging over two thetas- combining
   rules are AND, OR, XOR- XOR is just for fun.

wrapper2 = function(testpreds_i, testpreds_j, valpreds_i,
   valpreds_j, Ytest, Yval, thresh1, thresh2){
    testlabels_i = label(testpreds_i, thresh1)
    testlabels_j = label(testpreds_j, thresh2)
    vallabels_i = label(valpreds_i, thresh1)
    vallabels_j = label(valpreds_j, thresh2)
   #grab diversity metrics
    testcorr = correlation(testlabels_i, testlabels_j, Ytest)
    valcorr = correlation(vallabels_i, vallabels_j, Yval)
    testyule = yuleq(testlabels_i, testlabels_j, Ytest)
    valyule = yuleq(vallabels_i, vallabels_j, Yval)
    testdf = df(testlabels_i, testlabels_j, Ytest)
    valdf = df(vallabels_i, vallabels_j, Yval)
    testdisag = disagreement(testlabels_i, testlabels_j)
    valdisag = disagreement(vallabels_i, vallabels_j)
    testent = entropy(testlabels_i, testlabels_j, Ytest)
```

```
 valent = entropy(vallabels_i, vallabels_j, Yval)
 testKWV = KWV(testlabels_i, testlabels_j, Ytest)
 valKWV = KWV(vallabels_i, vallabels_j, Yval)
#do fusion on labels
 andtestlabels = testlabels_i & testlabels_j
 andvallabels = vallabels_i & vallabels_j
 ortestlabels = testlabels_i | testlabels_j
 orvallabels = vallabels_i | vallabels_j
 xortestlabels = (testlabels_i + testlabels_j)%%2
 xorvallabels = (vallabels_i + vallabels_j)%%2

#do accuracy on fused results
 andtestacc = accuracy(andtestlabels, Ytest)
 andvalacc = accuracy(andvallabels, Yval)
 ortestacc = accuracy(ortestlabels, Ytest)
 orvalacc = accuracy(orvallabels, Yval)
 xortestacc = accuracy(xortestlabels, Ytest)
 xorvalacc = accuracy(xorvallabels, Yval)

 #create fused tp, fp, tn, fn
 andtesttp = tpr(andtestlabels, Ytest)
 andtestfp = fpr(andtestlabels, Ytest)
 andtesttn = tnr(andtestlabels, Ytest)
 andtestfn = fnr(andtestlabels, Ytest)
 andvaltp = tpr(andvallabels, Yval)
 andvalfp = fpr(andvallabels, Yval)
```

```
  andvaltn = tnr(andvallabels, Yval)
  andvalfn = fnr(andvallabels, Yval)
  ortesttp = tpr(ortestlabels, Ytest)
  ortestfp = fpr(ortestlabels, Ytest)
  ortesttn = tnr(ortestlabels, Ytest)
  ortestfn = fnr(ortestlabels, Ytest)
  orvaltp = tpr(orvallabels, Yval)
  orvalfp = fpr(orvallabels, Yval)
  orvaltn = tnr(orvallabels, Yval)
  orvalfn = fnr(orvallabels, Yval)
  xortesttp = tpr(xortestlabels, Ytest)
  xortestfp = fpr(xortestlabels, Ytest)
  xortesttn = tnr(xortestlabels, Ytest)
  xortestfn = fnr(xortestlabels, Ytest)
  xorvaltp = tpr(xorvallabels, Yval)
  xorvalfp = fpr(xorvallabels, Yval)
  xorvaltn = tnr(xorvallabels, Yval)
  xorvalfn = fnr(xorvallabels, Yval)


#return results
  return(c(andtestacc, ortestacc, xortestacc, testcorr,
     testyule, testdf, testdisag, testent, testKWV,
     andvalacc, orvalacc, xorvalacc, valcorr,
      valyule, valdf, valdisag, valent, valKWV, andtesttp,
         andtestfp, andtesttn, andtestfn, andvaltp, andvalfp,
          andvaltn, andvalfn, ortesttp,
      ortestfp, ortesttn, ortestfn, orvaltp, orvalfp, orvaltn
         , orvalfn, xortesttp, xortestfp, xortesttn,
         xortestfn, xorvaltp, xorvalfp, xorvaltn,
      xorvalfn))

}

wrapperF = function(preds_i, preds_j, theta_i, theta_j, truth
   )
{
        npreds_i = array(0, c(length(preds_i),2))
        npreds_j = array(0, c(length(preds_j),2))
        for(i in 1:length(preds_i)){
        npreds_i[i, 1] = max(0, (theta_i-preds_i[i])/theta_i)
        npreds_j[i, 1] = max(0, (theta_j-preds_j[i])/theta_j)
        npreds_i[i, 2] = max(0, (preds_i[i]-theta_i)/theta_i)
        npreds_j[i, 2] = max(0, (preds_j[i]-theta_j)/theta_j)
        }
        bempreds = npreds_i/2 + npreds_j/2
        prodpreds = npreds_i * npreds_j
        minpreds = array(0, c(length(preds_i), 2))
        maxpreds = array(0, c(length(preds_i), 2))
        for(i in 1:length(preds_i)){
```

```
        minpreds[i,1] = min(npreds_i[i,1], npreds_j[i,1])
        minpreds[i,2] = min(npreds_i[i,2], npreds_j[i,2])
        maxpreds[i,1] = max(npreds_i[i,1], npreds_j[i,1])
        maxpreds[i,2] = max(npreds_i[i,2], npreds_j[i,2])
        }
        bemlabels = bempreds[,2]>bempreds[,1]
        prodlabels = prodpreds[,2]>prodpreds[,1]
        minlabels = minpreds[,2]>minpreds[,1]
        maxlabels = maxpreds[,2]>maxpreds[,1]
        bemacc = accuracy(bemlabels, truth)
        prodacc = accuracy(prodlabels, truth)
        minacc = accuracy(minlabels, truth)
        maxacc = accuracy(maxlabels, truth)
        return(c(bemacc, prodacc, minacc, maxacc))
}

wrapperF2 = function(preds_i, preds_j, theta, truth){
        preds_i = cbind(1-preds_i, preds_i)
        preds_j = cbind(1-preds_j, preds_j)
        bempreds = preds_i/2 + preds_j/2
        prodpreds = preds_i * preds_j
        minpreds = array(0, c(nrow(preds_i), 2))
        maxpreds = array(0, c(nrow(preds_i), 2))
        for(i in 1:nrow(preds_i)){
        minpreds[i,1] = min(preds_i[i,1], preds_j[i,1])
        minpreds[i,2] = min(preds_i[i,2], preds_j[i,2])
        maxpreds[i,1] = max(preds_i[i,1], preds_j[i,1])
        maxpreds[i,2] = max(preds_i[i,2], preds_j[i,2])
        }
        bemlabels = bempreds[,2]>=theta
        prodlabels = prodpreds[,2]>=theta
        minlabels = minpreds[,2]>=theta
        maxlabels = maxpreds[,2]>=theta
        bemacc = accuracy(bemlabels, truth)
        prodacc = accuracy(prodlabels, truth)
        minacc = accuracy(minlabels, truth)
        maxacc = accuracy(maxlabels, truth)
        return(c(bemacc, prodacc, minacc, maxacc))
}

wrapperF3 = function(preds_i, preds_j, preds_k, theta_i,
   theta_j, theta_k, truth)
{
        npreds_i = array(0, c(length(preds_i),2))
        npreds_j = array(0, c(length(preds_j),2))
        npreds_k = array(0, c(length(preds_k),2))
        for(i in 1:length(preds_i)){
        npreds_i[i, 1] = max(0, (theta_i-preds_i[i])/theta_i)
        npreds_j[i, 1] = max(0, (theta_j-preds_j[i])/theta_j)
        npreds_k[i, 1] = max(0, (theta_k-preds_k[i])/theta_k)
        npreds_i[i, 2] = max(0, (preds_i[i]-theta_i)/theta_i)
        npreds_j[i, 2] = max(0, (preds_j[i]-theta_j)/theta_j)
        npreds_k[i, 2] = max(0, (preds_k[i]-theta_k)/theta_k)
        }
        bempreds = npreds_i/3 + npreds_j/3 + npreds_k/3
        prodpreds = npreds_i * npreds_j * npreds_k
        minpreds = array(0, c(length(preds_i), 2))
        maxpreds = array(0, c(length(preds_i), 2))
        for(i in 1:length(preds_i)){
        minpreds[i,1] = min(npreds_i[i,1], npreds_j[i,1],
           npreds_k[i,1])
        minpreds[i,2] = min(npreds_i[i,2], npreds_j[i,2],
           npreds_k[i,2])
        maxpreds[i,1] = max(npreds_i[i,1], npreds_j[i,1],
           npreds_k[i,1])
        maxpreds[i,2] = max(npreds_i[i,2], npreds_j[i,2],
           npreds_k[i,2])
        }
        bemlabels = bempreds[,2]>bempreds[,1]
        prodlabels = prodpreds[,2]>prodpreds[,1]
        minlabels = minpreds[,2]>minpreds[,1]
        maxlabels = maxpreds[,2]>maxpreds[,1]
        bemacc = accuracy(bemlabels, truth)
        prodacc = accuracy(prodlabels, truth)
        minacc = accuracy(minlabels, truth)
        maxacc = accuracy(maxlabels, truth)
        return(c(bemacc, prodacc, minacc, maxacc))
}
```

```
#create labels- binary only
label = function(probs, thresh) {
    labels = probs>=thresh
    return(labels)
    }

#calculate accuracy- binary only
accuracy = function(labels, truths) {
    correct = sum(labels==truths)
    total = length(labels==truths)
    acc=correct/total
    return(acc)
    }

tpr = function(labels, truths) {
    positives = sum(truths)
    truepositives = sum((labels==TRUE) & (truths==TRUE))
    return(truepositives/positives)
    }

fpr = function(labels, truths) {
    negatives = length(truths) - sum(truths)
    falsepositives = sum((labels==TRUE) & (truths==FALSE))
    return(falsepositives/negatives)
    }

tnr = function(labels, truths) {
    negatives = length(truths) - sum(truths)
    truenegatives = sum((labels==FALSE) & (truths==FALSE))
    return(truenegatives/negatives)
    }

fnr = function(labels, truths) {
    positives = sum(truths)
    falsenegatives = sum((labels==FALSE) & (truths==TRUE))
    return(falsenegatives/positives)
    }

#get class probabilities from kNN outputs
probs = function(kNNresults) {
    probabilities = attr(kNNresults, 'prob')
    posteriors = matrix(0,length(kNNresults),1)
    for(i in 1:length(kNNresults)){
        if(kNNresults[i]==1){
        posteriors[i]=probabilities[i]
        }else{
        posteriors[i]=1-probabilities[i]
        }
    }
    return(posteriors)
    }

#functions for calculating diversity metrics:

#disagreement
disagreement = function(labels_i, labels_j) {
    disagree = sum(labels_i != labels_j)
    N = length(labels_i)
    return(disagree/N)
    }

#correlation
correlation = function(labels_i, labels_j, truth) {
    N = length(truth)
    a = 0
    b = 0
    c = 0
    d = 0
    for(i in 1:N){
        a = a + (labels_i[i]==truth[i]) * (labels_j[i]==truth
            [i])
        b = b + (labels_i[i]==truth[i]) * (labels_j[i]!=truth
            [i])
        c = c + (labels_i[i]!=truth[i]) * (labels_j[i]==truth
            [i])
        d = d + (labels_i[i]!=truth[i]) * (labels_j[i]!=truth
            [i])
    }
    rho = (a*d-b*c)/sqrt((a+b)*(c+d)*(a+c)*(b+d))
        if(is.na(rho)){rho=1}
    return(rho)
    }

#Yule's Q
yuleq = function(labels_i, labels_j, truth) {
    N = length(truth)
    a = 0
    b = 0
    c = 0
    d = 0
    for(i in 1:N){
        a = a + (labels_i[i]==truth[i]) * (labels_j[i]==truth
            [i])
        b = b + (labels_i[i]==truth[i]) * (labels_j[i]!=truth
            [i])
        c = c + (labels_i[i]!=truth[i]) * (labels_j[i]==truth
            [i])
        d = d + (labels_i[i]!=truth[i]) * (labels_j[i]!=truth
            [i])
    }
    qstat = ((a*d)-(b*c))/((a*d)+(b*c))
    if(is.na(qstat)){qstat=1}
  return(qstat)
  }

#double fault
df = function(labels_i, labels_j, truth) {
  N = length(truth)
  a = 0
  b = 0
  c = 0
  d = 0
  for(i in 1:N){
    a = a + (labels_i[i]==truth[i]) * (labels_j[i]==truth
      [i])
    b = b + (labels_i[i]==truth[i]) * (labels_j[i]!=truth
      [i])
    c = c + (labels_i[i]!=truth[i]) * (labels_j[i]==truth
      [i])
    d = d + (labels_i[i]!=truth[i]) * (labels_j[i]!=truth
      [i])
  }
  return(d)
  }

#entropy- currently binary only implementation
entropy = function(labels_i, labels_j, truth) {
    N = length(truth)
    T = 2
    E = 0
    for(i in 1:N){
        funkyletter_i = (labels_i[i]!=truth[i]) + (labels_j[i
           ]!=truth[i])
        E = E + min(funkyletter_i, T-funkyletter_i)
    }
    E = E * (1/N) * (1/(T-T/2))
    return(E)
    }

#entropy- three classifier implementation
entropy3 = function(labels_i, labels_j, labels_k, truth) {
    N = length(truth)
    T = 3
    E = 0
    for(i in 1:N){
        funkyletter_i = (labels_i[i]!=truth[i]) + (labels_j[i
           ]!=truth[i]) + (labels_k[i]!=truth[i])
        E = E + min(funkyletter_i, T-funkyletter_i)
    }
    E = E * (1/N) # This part becomes 1 so is not needed-->
       (1/(T-ceiling(T/2)))
    return(E)
    }
```

```
#Kohavi-Wolpert variance-  binary only implementation
KWV = function(labels_i, labels_j, truth) {
    N = length(truth)
    T = 2
    KW = 0
    for(i in 1:N){
        funkyletter_i = (labels_i[i]!=truth[i]) + (labels_j[i
            ]!=truth[i])
        KW = KW + (funkyletter_i * (T-funkyletter_i))
    }
    KW = KW * (1/(N*T^2))
    return(KW)
    }

#KWV for three classifier combos can be found as the average
   of the pairwise diagreements mulitplied by 1/3.  See
   Kuncheva, 'Measures of diversity in classifier ensembles.'
```

Listing B.17. "Universal R code used in both experiments- Wrappers.R"

```
####### Wrapper for three classifier combos #######
wrapper3 = function(c1num, c2num, c3num, c1test, c2test,
   c3test, c1val, c2val, c3val, c1thresh, c2thresh, c3thresh,
    Ytest, Yval){

# Calculate individual classifier stuff #
c1testlab = label(c1test, c1thresh)
```

```
c2testlab = label(c2test, c2thresh)
c3testlab = label(c3test, c3thresh)

c1vallab = label(c1val, c1thresh)
c2vallab = label(c2val, c2thresh)
c3vallab = label(c3val, c3thresh)

c1testacc = accuracy(c1testlab, Ytest)
c2testacc = accuracy(c2testlab, Ytest)
c3testacc = accuracy(c3testlab, Ytest)

c1valacc = accuracy(c1vallab, Yval)
c2valacc = accuracy(c2vallab, Yval)
c3valacc = accuracy(c3vallab, Yval)

c1testtp = tpr(c1testlab, Ytest)
c2testtp = tpr(c2testlab, Ytest)
c3testtp = tpr(c3testlab, Ytest)

c1testfp = fpr(c1testlab, Ytest)
c2testfp = fpr(c2testlab, Ytest)
c3testfp = fpr(c3testlab, Ytest)

c1valtp = tpr(c1vallab, Yval)
c2valtp = tpr(c2vallab, Yval)
c3valtp = tpr(c3vallab, Yval)
```

```
c1valfp = fpr(c1vallab, Yval)
c2valfp = fpr(c2vallab, Yval)
c3valfp = fpr(c3vallab, Yval)

# Calculate pairwise diversity metrics #
c12testcorr = correlation(c1testlab, c2testlab, Ytest)
c13testcorr = correlation(c1testlab, c3testlab, Ytest)
c23testcorr = correlation(c2testlab, c3testlab, Ytest)

c12testyule = yuleq(c1testlab, c2testlab, Ytest)
c13testyule = yuleq(c1testlab, c3testlab, Ytest)
c23testyule = yuleq(c2testlab, c3testlab, Ytest)

c12testdf = df(c1testlab, c2testlab, Ytest)
c13testdf = df(c1testlab, c3testlab, Ytest)
c23testdf = df(c2testlab, c3testlab, Ytest)

c12testdisag = disagreement(c1testlab, c2testlab, Ytest)
c13testdisag = disagreement(c1testlab, c3testlab, Ytest)
c23testdisag = disagreement(c2testlab, c3testlab, Ytest)

c12testrmsd = sqrt(mean((c1test - c2test)^2))
c13testrmsd = sqrt(mean((c1test - c3test)^2))
c23testrmsd = sqrt(mean((c2test - c3test)^2))
```

```
c12valcorr = correlation(c1vallab, c2vallab, Yval)
c13valcorr = correlation(c1vallab, c3vallab, Yval)
c23valcorr = correlation(c2vallab, c3vallab, Yval)

c12valyule = yuleq(c1vallab, c2vallab, Yval)
c13valyule = yuleq(c1vallab, c3vallab, Yval)
c23valyule = yuleq(c2vallab, c3vallab, Yval)

c12valdf = df(c1vallab, c2vallab, Yval)
c13valdf = df(c1vallab, c3vallab, Yval)
c23valdf = df(c2vallab, c3vallab, Yval)

c12valdisag = disagreement(c1vallab, c2vallab, Yval)
c13valdisag = disagreement(c1vallab, c3vallab, Yval)
c23valdisag = disagreement(c2vallab, c3vallab, Yval)

c12valrmsd = sqrt(mean((c1val - c2val)^2))
c13valrmsd = sqrt(mean((c1val - c3val)^2))
c23valrmsd = sqrt(mean((c2val - c3val)^2))

# Calculate ensemble diversity metrics #
testcorr = (c12testcorr+c13testcorr+c23testcorr)/3
testyule = (c12testyule+c13testyule+c23testyule)/3
testdf = (c12testdf+c13testdf+c23testdf)/3
testdisag = (c12testdisag+c13testdisag+c23testdisag)/3
testrmsd = (c12testrmsd+c13testrmsd+c23testrmsd)/3
```

```
testent = entropy3(c1testlab, c2testlab, c3testlab, Ytest)
testKWV = KWV3(c1testlab, c2testlab, c3testlab, Ytest)

valcorr = (c12valcorr+c13valcorr+c23valcorr)/3
valyule = (c12valyule+c13valyule+c23valyule)/3
valdf = (c12valdf+c13valdf+c23valdf)/3
valdisag = (c12valdisag+c13valdisag+c23valdisag)/3
valrmsd = (c12valrmsd+c13valrmsd+c23valrmsd)/3
valent = entropy3(c1vallab, c2vallab, c3vallab, Yval)
valKWV = KWV3(c1vallab, c2vallab, c3vallab, Yval)

# Do label fusion #
MVOTEtestlab = ((c1testlab + c2testlab + c3testlab)>=2)
MVOTEvallab = ((c1vallab + c2vallab + c3vallab)>=2)

# Calculate scores for measurement level fusions #
c1testscores = array(0, c(length(c1test),2))
c2testscores = array(0, c(length(c2test),2))
c3testscores = array(0, c(length(c3test),2))
for(i in 1:length(c1test)){
c1testscores[i,1] = max(0, (c1thresh-c1test[i])/c1thresh)
c2testscores[i,1] = max(0, (c2thresh-c2test[i])/c2thresh)
c3testscores[i,1] = max(0, (c3thresh-c3test[i])/c3thresh)
c1testscores[i,2] = max(0, (c1test[i]-c1thresh)/(1-c1thresh))
c2testscores[i,2] = max(0, (c2test[i]-c2thresh)/(1-c2thresh))
c3testscores[i,2] = max(0, (c3test[i]-c3thresh)/(1-c3thresh))
}

c1valscores = array(0, c(length(c1val),2))
c2valscores = array(0, c(length(c2val),2))
c3valscores = array(0, c(length(c3val),2))
for(i in 1:length(c1val)){
c1valscores[i,1] = max(0, (c1thresh-c1val[i])/c1thresh)
c2valscores[i,1] = max(0, (c2thresh-c2val[i])/c2thresh)
c3valscores[i,1] = max(0, (c3thresh-c3val[i])/c3thresh)
c1valscores[i,2] = max(0, (c1val[i]-c1thresh)/(1-c1thresh))
c2valscores[i,2] = max(0, (c2val[i]-c2thresh)/(1-c2thresh))
c3valscores[i,2] = max(0, (c3val[i]-c3thresh)/(1-c3thresh))
}

# Calculate weights for GEM- use these with test AND val sets
    #
misfit1 = c1test - Ytest
misfit2 = c2test - Ytest
misfit3 = c3test - Ytest
misfitcor = cor(cbind(misfit1, misfit2, misfit3))

#check that correlation matrix isn't broken from 100%
   accurate classifiers (it happens)
#basically if there is a 100% (or 0%) accurate classifier its
    misfit function will have
```

```
#no standard deviation and thus no correlation.  To be able
   to calculate an inverse, we
#assign a correlation of 0- while not technically true, zero
   fits nice because it is
#neither negative or positive.
misfitcor[is.na(misfitcor)]=0
Cinv = ginv(misfitcor)

a1 = sum(Cinv[,1])/sum(Cinv) #<- coefficient for classifier 1
a2 = sum(Cinv[,2])/sum(Cinv) #<- coefficient for classifier 2
a3 = sum(Cinv[,3])/sum(Cinv) #<- coefficient for classifier 3

# Do measurement level fusions #
BEMtest = (c1testscores + c2testscores + c3testscores)/3
GEMtest = a1 * c1testscores + a2 * c2testscores + a3 *
   c3testscores
PROtest = c1testscores * c2testscores * c3testscores
MINtest = array(0, c(length(c1test), 2))
MAXtest = array(0, c(length(c1test), 2))
for(i in 1:length(c1test)){
MINtest[i,1] = min(c1testscores[i,1], c2testscores[i,1],
   c3testscores[i,1])
MINtest[i,2] = min(c1testscores[i,2], c2testscores[i,2],
   c3testscores[i,2])
MAXtest[i,1] = max(c1testscores[i,1], c2testscores[i,1],
   c3testscores[i,1])
MAXtest[i,2] = max(c1testscores[i,2], c2testscores[i,2],
    c3testscores[i,2])
}
BEMtestlab = (BEMtest[,2] > BEMtest[,1])
GEMtestlab = (GEMtest[,2] > GEMtest[,1])
PROtestlab = (PROtest[,2] > PROtest[,1])
MINtestlab = (MINtest[,2] > MINtest[,1])
MAXtestlab = (MAXtest[,2] > MAXtest[,1])

BEMval = (c1valscores + c2valscores + c3valscores)/3
GEMval = a1 * c1valscores + a2 * c2valscores + a3 *
    c3valscores
PROval = c1valscores * c2valscores * c3valscores
MINval = array(0, c(length(c1val), 2))
MAXval = array(0, c(length(c1val), 2))
for(i in 1:length(c1val)){
MINval[i,1] = min(c1valscores[i,1], c2valscores[i,1],
    c3valscores[i,1])
MINval[i,2] = min(c1valscores[i,2], c2valscores[i,2],
    c3valscores[i,2])
MAXval[i,1] = max(c1valscores[i,1], c2valscores[i,1],
    c3valscores[i,1])
MAXval[i,2] = max(c1valscores[i,2], c2valscores[i,2],
    c3valscores[i,2])
}
BEMvallab = (BEMval[,2] > BEMval[,1])
```

```
GEMvallab = (GEMval[,2] > GEMval[,1])
PROvallab = (PROval[,2] > PROval[,1])
MINvallab = (MINval[,2] > MINval[,1])
MAXvallab = (MAXval[,2] > MAXval[,1])

# Calculate ensemble stuff #
MVOTEtestacc = accuracy(MVOTEtestlab, Ytest)
BEMtestacc = accuracy(BEMtestlab, Ytest)
GEMtestacc = accuracy(GEMtestlab, Ytest)
PROtestacc = accuracy(PROtestlab, Ytest)
MINtestacc = accuracy(MINtestlab, Ytest)
MAXtestacc = accuracy(MAXtestlab, Ytest)

MVOTEtesttp = tpr(MVOTEtestlab, Ytest)
BEMtesttp = tpr(BEMtestlab, Ytest)
GEMtesttp = tpr(GEMtestlab, Ytest)
PROtesttp = tpr(PROtestlab, Ytest)
MINtesttp = tpr(MINtestlab, Ytest)
MAXtesttp = tpr(MAXtestlab, Ytest)

MVOTEtestfp = fpr(MVOTEtestlab, Ytest)
BEMtestfp = fpr(BEMtestlab, Ytest)
GEMtestfp = fpr(GEMtestlab, Ytest)
PROtestfp = fpr(PROtestlab, Ytest)
MINtestfp = fpr(MINtestlab, Ytest)
MAXtestfp = fpr(MAXtestlab, Ytest)
```

```
MVOTEvalacc = accuracy(MVOTEvallab, Yval)
BEMvalacc = accuracy(BEMvallab, Yval)
GEMvalacc = accuracy(GEMvallab, Yval)
PROvalacc = accuracy(PROvallab, Yval)
MINvalacc = accuracy(MINvallab, Yval)
MAXvalacc = accuracy(MAXvallab, Yval)

MVOTEvaltp = tpr(MVOTEvallab, Yval)
BEMvaltp = tpr(BEMvallab, Yval)
GEMvaltp = tpr(GEMvallab, Yval)
PROvaltp = tpr(PROvallab, Yval)
MINvaltp = tpr(MINvallab, Yval)
MAXvaltp = tpr(MAXvallab, Yval)

MVOTEvalfp = fpr(MVOTEvallab, Yval)
BEMvalfp = fpr(BEMvallab, Yval)
GEMvalfp = fpr(GEMvallab, Yval)
PROvalfp = fpr(PROvallab, Yval)
MINvalfp = fpr(MINvallab, Yval)
MAXvalfp = fpr(MAXvallab, Yval)

# Return values as a list #
return(c(c1num, c2num, c3num, c1thresh, c2thresh, c3thresh,
   c1testacc, c2testacc, c3testacc, c1valacc, c2valacc,
   c3valacc, c1testtp, c2testtp, c3testtp, c1testfp, c2testfp
```

```
    , c3testfp , c1valtp , c2valtp , c3valtp , c1valfp , c2valfp ,
    c3valfp , testcorr , testyule , testdf , testdisag , testrmsd ,
    testent , testKWV, valcorr , valyule , valdf , valdisag ,
    valrmsd , valent , valKWV, MVOTEtestacc, MVOTEtesttp,
    MVOTEtestfp, BEMtestacc, BEMtesttp, BEMtestfp, GEMtestacc,
     GEMtesttp, GEMtestfp, PROtestacc, PROtesttp, PROtestfp,
    MINtestacc, MINtesttp, MINtestfp, MAXtestacc, MAXtesttp,
    MAXtestfp, MVOTEvalacc, MVOTEvaltp, MVOTEvalfp, BEMvalacc,
     BEMvaltp, BEMvalfp, GEMvalacc, GEMvaltp, GEMvalfp,
    PROvalacc, PROvaltp, PROvalfp, MINvalacc, MINvaltp,
    MINvalfp, MAXvalacc, MAXvaltp, MAXvalfp))
}

wrapper1 = function(c1num, c2num, c3num, c1test, c2test,
    c3test, c1val, c2val, c3val, thresh, Ytest, Yval){

# Calculate weights for GEM- use these with test AND val sets
     #
misfit1 = c1test - Ytest
misfit2 = c2test - Ytest
misfit3 = c3test - Ytest
misfitcor = cor(cbind(misfit1, misfit2, misfit3))

#check that correlation matrix isn't broken from 100%
    accurate classifiers (it happens)
```

```
#basically if there is a 100% (or 0%) accurate classifier its
    misfit function will have
#no standard deviation and thus no correlation.  To be able
   to calculate an inverse, we
#assign a correlation of 0— while not technically true, zero
   fits nice because it is
#neither negative or positive.
misfitcor[is.na(misfitcor)]=0
Cinv = ginv(misfitcor)

a1 = sum(Cinv[,1])/sum(Cinv) #<- coefficient for classifier 1
a2 = sum(Cinv[,2])/sum(Cinv) #<- coefficient for classifier 2
a3 = sum(Cinv[,3])/sum(Cinv) #<- coefficient for classifier 3

# Do measurement level fusions #
BEMtest = (c1test + c2test + c3test)/3
GEMtest = a1 * c1test + a2 * c2test + a3 * c3test
PROtest = c1test * c2test * c3test
MINtest = array(0, length(c1test))
MAXtest = array(0, length(c1test))
for(i in 1:length(c1test)){
MINtest[i] = min(c1test, c2test, c3test)
MAXtest[i] = max(c1test, c2test, c3test)
}
BEMtestlab = label(BEMtest, thresh)
GEMtestlab = label(GEMtest, thresh)
```

```
PROtestlab = label(PROtest, thresh)
MINtestlab = label(MINtest, thresh)
MAXtestlab = label(MAXtest, thresh)

BEMval = (c1val + c2val + c3val)/3
GEMval = a1 * c1val + a2 * c2val + a3 * c3val
PROval = c1val * c2val * c3val
MINval = array(0, length(c1val))
MAXval = array(0, length(c1val))
for(i in 1:length(c1val)){
MINval[i] = min(c1val, c2val, c3val)
MAXval[i] = max(c1val, c2val, c3val)
}
BEMvallab = label(BEMval, thresh)
GEMvallab = label(GEMval, thresh)
PROvallab = label(PROval, thresh)
MINvallab = label(MINval, thresh)
MAXvallab = label(MAXval, thresh)

# Calculate ensemble stuff #
BEMtestacc = accuracy(BEMtestlab, Ytest)
GEMtestacc = accuracy(GEMtestlab, Ytest)
PROtestacc = accuracy(PROtestlab, Ytest)
MINtestacc = accuracy(MINtestlab, Ytest)
MAXtestacc = accuracy(MAXtestlab, Ytest)
```

```
BEMvalacc = accuracy(BEMvallab, Yval)
GEMvalacc = accuracy(GEMvallab, Yval)
PROvalacc = accuracy(PROvallab, Yval)
MINvalacc = accuracy(MINvallab, Yval)
MAXvalacc = accuracy(MAXvallab, Yval)

return(c(c1num, c2num, c3num, thresh, BEMtestacc, GEMtestacc,
    PROtestacc, MINtestacc, MAXtestacc, BEMvalacc, GEMvalacc,
    PROvalacc, MINtestacc, MAXtestacc))
}
```

Appendix C. MATLAB code

All of the MATLAB code is posted here. There are two files, one for the Monte Carlo resampling experiment, and one for the main ensemble creation experiment. They are each initialized by a command embedded in the MatlabExportBootstrap.R and MatlabExport.R files ran from within R. When the command is called, R starts up MATLAB, runs the commands, and then closes MATLAB and continues with the rest of the code within R.

Listing C.1. "MATLAB code for Monte Carlo resampling experiment"

```
%enter directory where the input files are stored
cd '~/ThesisData/MATLABinputfiles';

filestring=[filename 'Bootstrap.mat'];
load(filestring);

for i = 1:30
%create PNN and RBF networks
eval(['PNN',int2str(i),'␣=␣newpnn(Xtrain',int2str(i),''',␣
    ind2vec(Ytrain',int2str(i),'''+1),2);'])
eval(['RBF',int2str(i),'␣=␣newrb(Xtrain',int2str(i),''',␣
    ind2vec(Ytrain',int2str(i),'''+1),␣0,␣2,␣40,␣5);'])

%change layers over to softmax for posterior probabilities-
    originally,
%MATLAB's PNN implementation has a competitive layer that
    only outputs 0 or
```

```
%1, and the RBF implementation has a pure-linear layer that
    allows negative
%values.  Changing these layers to a 'softmax' creates
    outputs that are
%{0,1} and sum to 1, this can be interpreted as class
    probabilities

eval(['PNN',int2str(i),'.layers{2}.transferFcn = ''softmax'';
    '])
eval(['RBF',int2str(i),'.layers{2}.transferFcn = ''softmax'';
    '])

%get posterior probabilities for test and validation sets
eval(['PNN',int2str(i),'testout = PNN',int2str(i),'(Xtest',
    int2str(i),''');'])
eval(['RBF',int2str(i),'testout = RBF',int2str(i),'(Xtest',
    int2str(i),''');'])
eval(['PNN',int2str(i),'valout = PNN',int2str(i),'(Xval',
    int2str(i),''');'])
eval(['RBF',int2str(i),'valout = RBF',int2str(i),'(Xval',
    int2str(i),''');'])

%only really care about the probabilities for the "positive"
    class... so we
%get rid of the first row of probabilities, and also make it
    a column
```

```
%vector for easier importing back to R.
eval(['PNN',int2str(i),'testout = PNN',int2str(i),'testout
    (2,:)'';'])
eval(['RBF',int2str(i),'testout = RBF',int2str(i),'testout
    (2,:)'';'])
eval(['PNN',int2str(i),'valout = PNN',int2str(i),'valout(2,:)
    '';'])
eval(['RBF',int2str(i),'valout = RBF',int2str(i),'valout(2,:)
    '';'])

end



%write output files, time to head back into the R script
cd ~/ThesisData/MATLABoutputfiles
save(filestring, '-v4');

%switch back to top directory
cd ~/ThesisData
```

.

Listing C.2. "MATLAB code for ensemble creation experiment"

```
%enter directory where the input files are stored
cd '~/ThesisData/MATLABinputfiles';

%load data- disregard string data (header) at top of file, we
     only want the
```

```
%numerical data
Xtest = importdata(strcat(filename, '-Xtest.csv'));
Xtest = Xtest.data;
Xval = importdata(strcat(filename, '-Xval.csv'));
Xval = Xval.data;
Xtrain = importdata(strcat(filename, '-Xtrain.csv'));
Xtrain = Xtrain.data;
Ytrain = importdata(strcat(filename, '-Ytrain.csv'));
Ytrain = Ytrain.data;

%create PNN and RBF networks
PNN = newpnn(Xtrain', ind2vec(Ytrain'+1),2);
RBF = newrb(Xtrain', ind2vec(Ytrain'+1), 0, 2, 40, 5);

%change layers over to softmax for posterior probabilities-
    originally,
%MATLAB's PNN implementation has a competitive layer that
    only outputs 0 or
%1, and the RBF implementation has a pure-linear layer that
    allows negative
%values.  Changing these layers to a 'softmax' creates
    outputs that are
%{0,1} and sum to 1, this can be interpreted as class
    probabilities
PNN.layers{2}.transferFcn = 'softmax';
RBF.layers{2}.transferFcn = 'softmax';
```

```
%get posterior probabilities for test and validation sets
PNNtestout = PNN(Xtest');
RBFtestout = RBF(Xtest');
PNNvalout = PNN(Xval');
RBFvalout = RBF(Xval');


%only really care about the probabilities for the "positive"
    class... so we
%get rid of the first row of probabilities, and also make it
    a column
%vector for easier importing back to R.
PNNtestout = PNNtestout(2,:)';
RBFtestout = RBFtestout(2,:)';
PNNvalout = PNNvalout(2,:)';
RBFvalout = RBFvalout(2,:)';


%write output files, time to head back into the R script
cd ~/ThesisData/MATLABoutputfiles
csvwrite(strcat(filename, '-PNNtestpreds.csv'), PNNtestout);
csvwrite(strcat(filename, '-RBFtestpreds.csv'), RBFtestout);
csvwrite(strcat(filename, '-PNNvalpreds.csv'), PNNvalout);
csvwrite(strcat(filename, '-RBFvalpreds.csv'), RBFvalout);
```

```
%switch back to top directory
cd ~/ThesisData
```

Appendix D. Quad Chart

The Quad Chart for this research is found below.

The Effectiveness of Using Diversity in Selecting Multiple Classifier Systems

2Lt Harris Butler
Advisor: LtCol Mark A. Friend, PhD
Department of Operational Sciences (ENS)
Air Force Institute of Technology

Introduction

Classification is the process of grouping observations, called exemplars, into different categories, called classes.

Many algorithms exist to automate classification, but in recent years progress on individual classifiers has slowed. Research in recent years has focused on combining the outputs of multiple classification algorithms to create more accurate classifications, the combined outputs create a Multiple Classifier System (MCS).

There is speculation that the maximum accuracy occurs when the individual classifiers in the MCS have different outputs. The difference in the outputs of the individual classifiers is called diversity. Many researchers believe that maximizing diversity will create more accurate MCSs.

Research suggests that there may be a relationship between accuracy and diversity, but so far the evidence has not been conclusive.

Motivation

Computer systems all over the world use classification. Banks, hospitals, and even the Air Force use classification. Improving the accuracy of the classifiers in these systems increases the effectiveness of these operations.

Problem Statement

- Is there a relationship between the accuracy and diversity of an MCS?
- If there is a relationship, can it be exploited to build more accurate and robust MCSs?

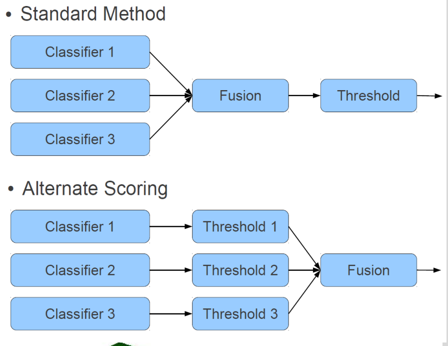
Alternate Scoring Technique

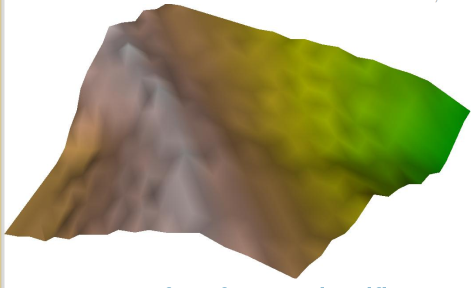
Introduced an alternate scoring technique that allowed us to look at a broader classification space which provides more data for an in depth look at diversity

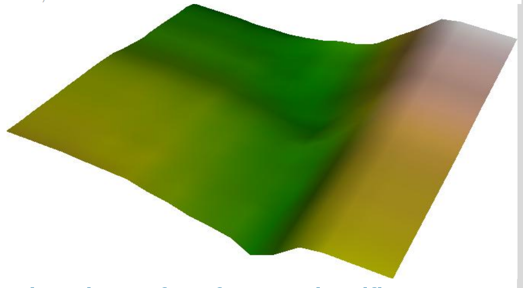
Take classifier t's class probability of class 1, $d_{t,1}$, and re-score based on a selected classification threshold, θ.

$$d^*_{t,1} = max(0, \frac{d_{t,1} - \theta}{1 - \theta})$$
$$d^*_{t,0} = max(0, \frac{\theta - d_{t,0}}{\theta})$$

- This creates a new class score that is still in the range [0,1]
- The scores no longer add up to 1 like class probabilities
- Classification is performed by comparing $d^*_{t,1} < or \geq d^*_{t,0}$




Accuracy surface for two classifiers


Diversity surface for two classifiers

Diversity Metric	Within Set Correlation	Between Sets Correlation
Corr	0.023	-0.035
DF	0.352	0.238
Disag	-0.106	-0.124
Ent	-0.106	-0.124
KWV	-0.106	-0.124
YuleQ	0.001	-0.042

There is very low correlation between the popular diversity metrics and the accuracy of an MCS

	Accuracy		Diversity					
Fusion Technique	Ensemble	Individual	Corr	Yule	DF	Disag	Ent	KWV
BEM	94%	95%	90%	90%	93%	91%	91%	91%
GEM	93%	93%	87%	89%	91%	86%	86%	86%
MAX	95%	95%	90%	88%	91%	91%	91%	91%
MIN	95%	93%	78%	80%	78%	66%	66%	66%
MVOTE	93%	95%	86%	86%	93%	83%	83%	83%
PROD	95%	93%	78%	80%	78%	66%	66%	66%

Using the accuracy of a test set as the criteria for selecting classifiers for an MCS performs much better than using the diversity of the test set

Results

There is very low correlation between accuracy and diversity of an MCS

Regression results show that there is a statistically significant relationship between accuracy and diversity, but it is very small

Accuracy is a much better criteria to use when selecting classifiers to build an MCS

Conclusions

- There is a statistically significant relationship between the accuracy and diversity of an MCS, but it is small and hard to characterize
- Because the relationship is small it is not of any practical use for selecting classifiers to build an MCS

Impacts/Contributions

- Introduced an alternate scoring technique that makes it possible to investigate diversity further
- Confirmed much prior research indicating that there is a relationship between accuracy and diversity, but it is still not clearly defined

Collaboration

AFIT Sensor Fusion Lab

AIR FORCE INSTITUTE of TECHNOLOGY

Bibliography

[1] Aksela, Matti and Jorma Laaksonen. “Using diversity of errors for selecting members of a committee classifier”, *Pattern Recognition*, 39:608–623, 2006.

[2] Atiya, Amir F. “Estimating the Posterior Probabilities Using the K-Nearest Neighbor Rule”, July 2004. Dept Computer Engineering, Cairo University.

[3] Brown, Gavin and Ludmila I. Kuncheva. “”Good” and ”Bad” Diversity in Majority Vote Ensembles.” Neamat El Gayar, Josef Kittler, and Fabio Roli (editors), *MCS*, volume 5997 of *Lecture Notes in Computer Science*, 124–133. Springer, 2010.

[4] Canuto, Anne M.P., Marjory C.C. Abreu, Lucas de Melo Oliveira, Joao C. Xavier Jr., and Araken de M. Santos. “Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles”, *Pattern Recognition Letters*, 28:472–486, 2007.

[5] Dimitriadou, Evgenia, Kurt Hornik, Friedrich Leisch, David Meyer, and Andreas Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2011. URL `http://CRAN.R-project.org/package=e1071`. R package version 1.6.

[6] Elter, M., R. Schulz-Wendtland, and T. Wittenberg. “The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process”, *Medical Physics*, 34(11):4164–4172, 2007.

[7] Evett, Ian W. and Ernest J. Spiehler. “Rule Induction in Forensic Science”. Central Research Establishment, Home Office Forensic Science Service, Aldermaston, Reading, Berkshire RG7 4PN, September 1987.

[8] Fawcett, Tom. *ROC Graphs: Notes and Practical Considerations for Researchers*. Technical report, HP Laboratories, 2004.

[9] Fisher, R. A. “The use of multiple measurements in taxonomic problems”, *Annual Eugenics*, 7:179–188, 1936.

[10] Flach, Peter A. “Tutorial: The many faces of ROC analysis in machine learning”.

[11] Forsyth, Richard S. “BUPA Medical Research Ltd- liver disorders”. PC/BEAGLE User’s guide, May 1990.

[12] Frank, A. and A. Asuncion. “UCI Machine Learning Repository”, 2010. URL `http://archive.ics.uci.edu/ml`.

[13] Gacquer, David, Veronique Delcroix, Franois Delmotte, and Sylvain Piechowiak. "On the Effectiveness of Diversity When Training Multiple Classifier Systems." Claudio Sossai and Gaetano Chemello (editors), *ECSQARU*, volume 5590 of *Lecture Notes in Computer Science*, 493–504. Springer, 2009.

[14] Haberman, S. J. "Generalized Residuals for Log-Linear Models". *Proceedings of the 9th International Biometrics Conference*. 1976.

[15] Hadjitodorov, Stefan Todorov, Ludmila I. Kuncheva, and Ludmila P. Todorova. "Moderate diversity for better cluster ensembles.", *Information Fusion*, 7(3):264–275, 2006.

[16] Hastie, T., R. Tibshirani, J. Friedman, and J. Franklin. *The elements of statistical learning: data mining, inference and prediction*, volume 27. Springer, 2005.

[17] Hill, J.M., M.E. Oxley, and K.W.; Bauer. "Receiver operating characteristic curves and fusion of multiple classifiers." *Proceedings of the Sixth International Conference of Information Fusion, 2003*, 815–822. IEEE, 2003.

[18] Hopkins, Mark, Erik Reeber, George Forman, and Jaap Suermondt. *SPAM E-mail Database*. Technical report, Hewlett-Packard Labs, 1999.

[19] Hornik, K., M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators", *Neural Networks*, 2:356–366, 1989.

[20] Kuncheva, Ludmila and Roumen K. Kounchev. "Generating classifier outputs of fixed accuracy and diversity.", *Pattern Recognition Letters*, 23(5):593–600, 2002.

[21] Kuncheva, Ludmila I. "That Elusive Diversity in Classifier Ensembles". *Lecture Notes in Computer Science*, 1126–1138. 2003.

[22] Kuncheva, Ludmila I., James C. Bezdek, and Robert P. W. Duin. "Decision templates for multiple classifier fusion: an experimental comparison", *Pattern Recognition*, 34:299–314, 2001.

[23] Kuncheva, Ludmila I. and Christopher J. Whitaker. "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy.", *Machine Learning*, 51(2):181–207, 2003.

[24] Kurgan, L. A., K. J. Cios, R. Tadeusiewicz, M. Ogiela, and L. S. Goodenday. "Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis", *Artificial Intelligence in Medicine*, 23(2):149–169, 2001.

[25] Little, M. A., P. E. McSharry, S. J. Roberts, D. A. E. Costello, and I. M. Moroz. "Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection", *BioMedical Engineering OnLine*, 6(23):0, 2007.

[26] Lu, Y. "Knowledge Integration in a Multiple Classifier System", *Applied Intelligence*, 6(2):75–86, 1996.

[27] Macskassy, Sofus A., Foster Provost, and Saharon Rosset. "ROC Confidence Bands: An Empirical Evaluation". *In: Proceedings of the Twenty-Second International Conference on Machine Learning*, 537–544. ACM, 2005.

[28] MATLAB. *version 7.13.0.564 (R2011b)*. The MathWorks Inc., Natick, Massachusetts, 2011.

[29] Messer, Adam J. *Contextual Detection of Anomalies Within Hyperspectral Images*. Master's thesis, Air Force Institute of Technology, 2011.

[30] Michalski, R. S., I. Mozetic, J. Hong, and N. Lavrac. "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains". *Proceedings of the Fifth National Conference on Artificial Intelligence*. 1986.

[31] Perrone, Michael P. and Leaon N. Cooper. "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks". 126–142. Chapman and Hall, 1993.

[32] Polikar, Robi. "Ensemble Based Systems in Decision Making", *IEEE Circuits and Systems*, 21–45, 2006.

[33] Quinlan. "Simplifying decision trees", *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[34] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

[35] Rogova, G. "Combining the Results of Several Neural Network Classifiers", *Neural Networks*, 7(5):777–781, 1994.

[36] Roli, Fabio. "Tutorial: Fusion of Multiple Pattern Classifiers". AI*IA, 2003.

[37] Ruta, Dymitr and Bogdan Gabrys. "An Overview of Classifier Fusion Methods", 2000.

[38] Ruta, Dymitr and Bogdan Gabrys. "Analysis of the Correlation between Majority Voting Error and the Diversity Measures in Multiple Classifier Systems", 2001.

[39] Shipp, Catherine A. and Ludmila Kuncheva. "Relationships between combination methods and measures of diversity in combining classifiers.", *Information Fusion*, 3(2):135–148, 2002.

[40] Siegler, R. S. "Three Aspects of Cognitive Development", *Cognitive Psychology*, 8:481–520, 1976.

[41] Smith, J. W., J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. "Pima Indians Diabetes Dataset". *Proceedings of the Symposium on Computer Applications and Medical Care*, 261–265. 1988.

[42] Street, W.N., W.H. Wolberg, and O.L. Mangasarian. "Nuclear feature extraction for breast tumor diagnosis", *International Symposium on Electronic Imaging: Science and Technology*, 1905:861–870, 1993.

[43] Venables, W. N. and B. D. Ripley. *Modern Applied Statistics with S*, fourth edition). Springer, New York, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. ISBN 0-387-95457-0.

[44] Windeatt, Terry. "Diversity measures for multiple classifier system analysis and design.", *Information Fusion*, 6(1):21–36, 2005.

[45] Wolpert, David H. "The supervised learning no-free-lunch Theorems". *In Proc. 6th Online World Conference on Soft Computing in Industrial Applications*, 25–42. 2001.

[46] Yeh, I-Cheng, King-Jang Yang, and Tao-Ming Ting. "Knowledge discovery on RFM model using Bernoulli sequence", *Expert Systems with Applications*, 36(3):5866–5871, 2008.

Vita

Harris Butler graduated from Circle High School in Towanda, Kansas in 2006 and attended the U.S. Air Force Academy for the next four years. He entered the Air Force in 2010 upon graduating with a Bachelor of Science in Operations Research. His first assignment was to complete the Master of Science in Operations Research program at AFIT. Upon graduating from AFIT he will proceed to Nellis AFB to work as an analyst for AFOTEC Det 6.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 22–03–2012	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sep 2011 – Mar 2012

4. TITLE AND SUBTITLE

The Effectiveness of Using Diversity
For Selecting Multiple Classifier Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Harris K. Butler, 2Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-OR-MS-ENS-12-05

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED; DISTRIBUTION STATEMENT A

13. SUPPLEMENTARY NOTES

14. ABSTRACT A wealth of approaches exists to perform classification of items of interest. The goal of fusion techniques is to exploit complementary approaches and merge the information provided by these methods to provide a solution superior than any single method. Associated with choosing a fusion algorithm is the choice of algorithm or algorithms that will be fused. This decision is most often referred to as ensemble selection. Historically classifier ensemble accuracy has been used to accomplish this task. More recently research has focused on creating and evaluating diversity metrics to more effectively select ensemble members. This research focuses on the use of diversity as an ensemble selection methodology and explores the relationship between ensemble accuracy and diversity. Using a wide range of classification data sets, classification methodologies, and fusion techniques it extends current diversity research by expanding classifier domains before employing fusion methodologies; this is made possible with a unique classification score algorithm developed for this purpose. Correlation and linear regression techniques determine the relationship between examined diversity metrics and accuracy is tenuous and optimal ensemble selection should be based on ensemble accuracy.

15. SUBJECT TERMS

Classification, Classifier Fusion, Diversity, Pattern Recognition

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			LtCol Mark A. Friend, PhD
U	U	U	UU	159	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636 x4624: Mark.Friend@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18