Title: Rank and Response Combination From Confusion Matrix Data

Author name: Dr. J.R. Parker

Address: Department of Computer science
University of Calgary
2500 University Dr. NW
Calgary, Alberta, Canada
T2N 1N4

403 220 6784 (Office)

403 284 4707 (FAX)

Abstract:

The use of prior behavior of a classifier, as measured by the confusion matrix, can yield useful information for merging multiple classifiers. In particular, response vectors can be estimated and a ranking of possible classes can be produced which can allow Borda type reconciliation methods to be applied. A combination of real data and the simulation of multiple classifiers is used to evaluate this idea, and to compare with eleven other classifier combination techniques. Millions of classifications were used in the evaluation.

Rank and Response Combination From Confusion Matrix Data

J.R. Parker
Laboratory for Computer Vision
Department of Computer Science
University of Calgary
Calgary, Alberta, CANADA

Abstract

The use of prior behavior of a classifier, as measured by the confusion matrix, can yield useful information for merging multiple classifiers. In particular, response vectors can be estimated and a ranking of possible classes can be produced which can allow Borda type reconciliation methods to be applied. A combination of real data and the simulation of multiple classifiers is used to evaluate this idea, and to compare with eleven other classifier combination techniques. Millions of classifications were used in the evaluation.

1. Introduction

In the process of designing vision systems and pattern analysis algorithms, it is natural that a variety of techniques be tried, with the overall goal of improving the rate at which patterns can be correctly classified. Many diverse algorithms each have strengths and weaknesses, good ideas and bad. One way to take advantage of this variety is to apply many methods to the same recognition task, and have a scheme to merge the results [1,5,7,9,18]; this should be successful over a wider range of inputs than would any individual method. Indeed, it is what will be demanded of such a multiple classifier system: it must have a higher overall recognition rate than any of the individual classifiers that make it up, or there is no point to the extra effort involved.

Composite classifiers can be made up of similar classifier types (e.g. a collection of neural networks) or of quite diverse types (non-homogeneous) often using different classification techniques on different features. In the latter case it is hoped to take advantage of the diversity in the classifier set to achieve an increase in robustness. Classifiers can differ in both the nature of the measurements used in the classification process and in the type of classification produced. A Type I classification is a simple statement of the class, a Type II classification is a ranked list of probable classes, and a Type III classification assigns probabilities to classes [19]. Classifier combination algorithms more usually use Type III or Type I classifications as input, and many don't allow mixed types.

It is, of course, possible to convert between types. What will be suggested is the use of ranked classifications as a common form, and the use of variations on the Borda count [2] to combine the ranks into a single output classification. The use of prior information about the classifiers, in the form of the confusion matrix, will be used to construct response vectors and ranks, and it will be shown that there are significant advantages in doing this. Section 2 describes the Borda count and some variations and its relationship to other techniques; section 3 describes a mechanism for creating ranked classifications from other types; and section 4 will give some results and comparisons with eleven other known combination techniques. The goal of this work is to provide a reliable and robust way to combine non-homogeneous classifiers without using any retraining. While a neural network may be one of the classifiers, it is not expected that all or most classifiers will be; in other words, the methods described are not specific to particular classifier types.

2. Ranks and Borda Combination

Relatively few classifiers yield ranked classes directly, and rank combination techniques have not found much favor in the literature. The basic idea is that a classifier determine a relative order for its output classes; for example, the input was most likely of class B, less likely to be class C, and even less likely to be of class A. This example would be output as a list: B C A. It is not necessary to know the degree to which class B is more likely than class C. Indeed, if this is known then the output is of Type III and it is possible to take advantage of this extra knowledge. It is also true that Type III classifications are most easily converted into ranks, merely by sorting the output in descending order by probability and using the corresponding classes as the ranked list.

The problem encountered when attempting to merge ranked responses is as follows: given M rankings, each having R choices (classes), which choice has the largest degree of support? For example, consider the following 3 classifier/4 class problem [14]:

C1: a b c d   C2: c a b d   C3: b d c a

This case has no majority winner; the first place classifications a, b and c each get one vote.The *Borda count* (also called *Borda's method of marks*) is an ancient scheme for resolving this kind of situation, in which each alternative is given a number of points depending on where in the ranking it has been placed. A selection is given no points for placing last, one point for placing next to last, and so on, up to R-1 points for placing first. In other words, the number of points (the weight) given to a selection is the number of classes below it in the ranking. In the above example, the Borda counts are:

C1: a b c d	C2: c a b d	C3: b d c a
3 2 1 0	3 2 1 0	3 2 1 0

So, the count for class a = 3+2+0 = 5, the count for b = 2+1+3 = 6, and the count for c = 1+3+1 = 5. The 'winner' (the most likely class assignment in this case) is the class having the largest Borda count, which in this case is class b.

In the general case, each potential class assignment i=1,2,...R receives some number v_{i1} of first place votes, some number v_{i2} of second place votes, and so on. These are combined to give a desirability index D_i for each class. The Borda method for computing D_i is [4]:

$$D_i = \sum_{j=1}^{R} W_j v_{ij} \qquad \textbf{(EQ 1)}$$

The multipliers W_j are, in this case, just the number of classes having a lower rank than class j, or simply the value R-j. The class with the largest numerical desirability index is assigned. Using the outputs of the M classifiers explicitly, given the ranks r_{ik} from classifier k on class i gives:

$$D_i = \sum_{k=1}^{M} (R - r_{ik}) \qquad \textbf{(EQ 2)}$$

The vector r_k is the *rank vector* for classifier k, where a rank of 1 is the highest (first place) and a rank of R is the lowest (last place). Thus, r_{ki} is the rank assigned by classifier k to class i.

One question that arises concerns whether combining ranked classifications is in any sense better than combining type I, single classifications. However, consider the following 5 classifier/3 class problem:

C1: a b c   C2: a b c   C3: a b c   C4: b c a   C5: b c a

The Borda counts are **a=6**, **b=7**, **c=2**, which selects b as the winner. However, a simple majority of the first place votes would have selected **a**. This presents a conflict with the simple majority rule, and has been considered to be a problem for centuries.

Behind the Borda count is the presumption that the second most likely classification is relatively near, in terms of likelihood or preference, to the best classification; its rank is only one away it. Consider a four-candidate vote and the result A B C D. The sum of the ranks is 6 (in general N(N-1)/2 for N candidates). Treating these as scores, A gets 3 and B gets 2; the difference (1) is 1/6 of the total, the same as the difference between B and C, and the difference between C and D. In other words, a Borda count assumes that the distance between each candidate, once sorted, is the same, a presumption of uniformity.

This uniformity assumption is often flawed in the case of classifiers, although it may be the best that can be done for elections, the domain for which the Borda count was devised. Using prior information it is possible to more accurately estimate the relative distances between the ranked classifications and use these distances to calculate better weights for resolving the rankings in a Borda fashion [6]. Neural network output values can also be used for this purpose.

What is being proposed is a variable weighting of the ranked items. One suggestion is to use the measured properties of the classifier directly to assign a value to each rank position. The ranking of classes was produced from a set of measurements of a target. Without loss of generality, let f_{ik} be a numerical value actually computed by classifier k for class i, associated with the rank r_{ik}. Thus if $r_{ik}>r_{jk}$ then $f_{ik}<f_{jk}$; that is, larger values of f_{ik} indicate higher (= better = smaller numerically) ranks. The value of i for which $r_{ik}=1$ has the largest value of f_{ik}. The weights assigned to classes next to each other in rank, say, r_{ik} and $r_{i+1,k}$ should be related to f_{ik} and $f_{i+1,k}$. For classifier k, the vector $\bar{f}_k$ is called the *response vector.*

One rather obvious weight is the value f_{ik} itself. The rule for combination in this case would be

$$D_i = \sum_{k=1}^{M} f_{ik} \qquad \textbf{(EQ 3)}$$

in this case taking the largest value of D_i as indicating the class assignment. If the values of f are probabilities, then Equation 3 is the so-called *sum rule* [10] in the case where all classes have the same representation. This is related to the min rule, the max rule, and the median rule:

$$\begin{array}{ll} \text{max rule:} & \max_{k=1}^{m} \max_{i=1}^{R} f_{ik} \\ \text{min rule:} & \max_{k=1}^{m} \min_{i=1}^{R} f_{ik} \\ \text{median rule:} & \max_{k=1}^{m} \operatorname{median}_{i=1}^{R} f_{ik} \end{array} \qquad \textbf{(EQ 4)}$$

The more conservative *product rule* also belongs in this group:

$$\max_{k=1}^{m} \prod_{i=1}^{R} f_{ik} \qquad \textbf{(EQ 5)}$$

There is no compelling reason to use all of the classifiers, either, and some reason to favor the discarding of outliers. In particular, Tumer [15,16] suggests using only ranks M1 through M2 for some 1<=M1<=M2<=R. This he refers to as trimmed means, and it discards the highest and lowest ranked classifiers. It is, simply stated:

$$D_i = \frac{1}{M1-M2+1} \sum_{k=M1}^{M2} f_{ik} \qquad \textbf{(EQ 6)}$$

In the extreme this becomes the median rule. There may also be some value in considering only the extreme values (largest and smallest), such as:

$$D_i = \frac{1}{2}(f_{i1} + f_{iR}) \qquad \textbf{(EQ 7)}$$

This is called the *spread* combiner.

Another idea is to ignore the specific probabilities and assign simple non-uniform values to the ranked items. These weights could still be based on the typical observed distances between classes in a given classification, or could be constructed to achieve a specific goal. For example, again using the 5 classifier/3 class problem above, recall that the problem was that there was a conflict between the majority vote and the Borda count. Now use a weight of R-1 on the first ranked class, (R-2)*w on the second, and (R-3)*w^2 (=0) on the third. It becomes possible to find a weight w such that the winner found by summing the weights over all classifiers is the majority winner. In this latter instance the weight would be w=0.67; this use of constant weights is a generalization of the standard Borda count, referred to as *wBorda* in further discussion.

Finally, there are methods for minimizing some of the problems with the standard Borda count, the most obvious of which is the fact that it sometimes fails to find the majority winner. This has been discussed previously [11] in the context of handprinted digits. The use of the Borda count [2], Condorcet criterion [3], and the Black[14] scheme for combining these, were used to produce a combined classifier of five components that improved the recognition rate from 96% to over 99%. Essentially, the Black scheme uses a Condorcet (pair-wise contest) winner if there is one, thus guaranteeing that the majority winner, if one exists, will be the overall winner. If the Condorcet method results in a tie, the Borda winner is chosen.

3. Ranks From Simple Classifications

The previous discussion of reconciliation of ranked classifications is of little practical use unless such ranks are available. Most classifiers yield Type I or Type III classifications, so the problem is one of converting from these into ranks, and the question that remains is whether anything is lost or gained by doing so. Xu [19] used a collection of four classifiers of handprinted digits, and explored the use of Bayesian methodology, Dempster-Shafer theory, and voting to implement a combined classifier that improved the overall recognition rate to 98.9% from 93.9%. It was suggested here that ranked classifications could be created from the confusion matrix (*a posteriori* probabilities of each classification) of a Type I classifier. The assumptions are that, first, the behavior of the classifier is known and is characterized in a confusion matrix and, second, that the prior behavior or the classifier is representative of its future behavior. The larger the data set on which the classifier has been tested, the more thoroughly will the second assumption be true.

A confusion matrix is a matrix in which the actual class or a datum under test is represented by the matrix row, and the classification of that particular datum is represented by the confusion matrix column. The element M[i][j] gives the number of times that a class i object was assigned to class j. The diagonal elements indicate correct classifications and, if the matrix is not normalized, the sum of row I is the total number of elements of class I that actually appeared in the data set. The columns of such a matrix can be used to convert from Type I to Type III classifications, which can then be sorted to yield the ranks.

Consider a classifier that produces only a single output class (Type I) and that has been trained and tested on many thousands of data elements. During the this process, the following confusion matrix was generated (assume there are four classes):

Table 1
Example confusion matrix

	Classified as Class A	Classified as Class B	Classified as Class C	Classified as Class D
Actual class A	5210	159	101	530
Actual class B	320	5090	111	479
Actual class C	110	28	5813	148
Actual class D	210	12	7	5771

Each row sums to 6000, which was the number of elements of each class in the data set. Now as an example, presume that this classifier issues a classification of A for a given input datum. From the first column of the confusion matrix, it can be seen that the most likely actual class is A, the second most likely class is B, followed by D, and finally C. This is a fair ranking of the possible classes based on the past history of the classifier. In other words, given a classification of A:

5210/5850 will be correct (class A)
320/5850 will actually be class B
210/5850 will actually be class D
110/5850 will actually be class C

This is the scheme suggested for converting simple classifications into ranks. It will be accurate to the extent that the classifier has been thoroughly trained.

Does this improve the results over those possible using only the simple classifications? Yes. For example, in an experiment using five classifiers to recognize handprinted digits, a majority vote of the simple classifiers gives an overall recognition rate of 99.6%. Creating ranks from the same classifications and using a Borda count for combination improves this by 0.3% to 99.9%. Xu[19] achieves better improvements.

4. Empirical Evaluation

One of the problems with the empirical evaluation of classifier combination algorithms is the need for vast amounts of data. In this case, the data consists of classifier output from many different classifiers and a great many individual classification tasks. This means that the input data, with ground truth, must exist, and the classifiers must be correctly implemented. Even if sufficient data can be found, it has certain pre-defined properties that can't be altered. A classifier has a given recognition rate for a given type of data, and it may not be possible to provide all desired combinations of characteristics. For example, a classifier combination algorithm may work well when all individual classifiers operate at over 90% recognition, or when the classifiers have nearly the same rates. It is important to be able to determine benefits and limitations, either from a theoretical basis or empirically.

For these reasons it was decided to use both real data and a classifier simulator. The latter would generate classifications having a specified correct rate for each class as specified by an input confusion matrix.

4. 1 Real Data

A small set of classifications was available from previous experiments on handprinted digits. It consisted of 2000 digits, each classified by five distinct schemes [11]. The first thousand digits were used for training, leaving the remainder for testing. A single file was created containing all classifications and one confusion matrix for each classifier, and this was used as data for each of the classifier combination methods. This will be called the *digits* data set. In addition, four data sets were selected from the UCI archive: *iris*, *vehicle*, *segment*, and *waveform*. For these data, classifications were obtained using the WEKA system[17] using five schemes: 1R [8], naive Bayes (BAY), a decision table (DT), a nearest 2-neighbor instance learner (IB2), and C4.5 [13]. In all cases the result is five sets of classifications which are to be combined. The basic error rates for each classifier on each set of data are summarized in Table 2.

Without further training or use of prior information, the best that can be done with simple classifications of this kind is to combine them using a vote. This will be the base for comparison with the other techniques; if, by generating ranks and using rank combining methods no improvement can be achieved, then the method fails. The majority vote results were compared against those arrived at using a simple Borda count, the sum rule combiner, wBorda, the median rule, the max and min rules, trimmed means, and the spread combiner.

In addition to these, two new test combiners were included. Random1 selects one of the five classifications at random from each classifier for each class, and selects the maximum value in the response vectors. Random2 is similar, but selects two distinct classifications at random and selects the maximum of the sum of the two response values. These provide a minimal level of achievement for the combiners; a successful combiner should perform better than Random1, and a good combiner should perform better than Random2. In the limit, Random_k as k->M becomes the sum rule.

It should be noted that all data sets except *digits* are based on multiple classifiers on the same features. The digits set contains five quite distinct means of classifying visual objects, whereas the others simply apply a different classifier to the same numeric data. The *digits* set should yield more robust classifications, as appears to be illustrated in Table 3.

Table 2
Error Rates for Single Classifiers (real data)

	Digits	Iris	Segment	Vehicle	Waveform
1R	4.9	4.65	39.37	50.64	46.39
BAY	5.1	7.75	20.41	56.74	19.63
DT	5.4	4.65	14.14	41.28	32.69
IB2	6.7	4.65	10.10	42.70	28.05
C4.5	4.6	6.20	7.32	36.88	25.79

4. 2 Simulated Data

As before, let there be R possible classes. A real classifier is given an input pattern and produces either a classification, which is an integer in the range 1..R, or is unable to classify the input pattern, a situation represented by the classification value -1. A simulated classifier is given the *a posteriori* probabilities of each possible classification, as represented by a confusion matrix, and the actual class of the input; it also produces a classification as before, but based solely on the probabilities in the confusion matrix.

Of course, the confusion matrix does not tell all that can be told about the nature of a classifier. For only one example, the misclassification of a class A object as class B may result from two different processes, for two quite distinct reasons. The joint probability distribution associated with these two processes may well vary as a function of time, as the nature of the input data changes. The complexity associated with this situation may well be manageable, but as a first approximation the view will be taken that a classifier can be considered to be a single, possibly composite, process having relatively simple measurable properties.

There are two ways that simulated data can be used to test the classifier combination methods. One is to generate classifications based on the actual, measured performance of a known set of classifiers. This is done using the confusion matrix of the known classifiers, and the result of the combination should be similar to that obtained using the real data. The other method is to create confusion matrices having specified properties. In this latter case, the sum of the diagonal elements represents the correct classification rate and the off-diagonal elements represent the errors that the classifier can make.

A simulation module [12] generates simulated classifications, and an evaluation module implements the classifier combination algorithms and measures the properties of the composite classifiers. The combination techniques under consideration were as before, and each evaluation consists of 1000 individual classifications from each of five simulated classifiers merged and compared with the known result. For each set of five simulated classifiers, 1000 evaluations were performed and the results accumulated. Thus, five million simulated classifications were performed for each set of generated classification matrices. This was repeated over a specified range of recognition rates, using the same set of classifications for each combination method at each step so that they may be fairly evaluated.

Table 3
Error Rates for Combination Algorithms (real data)

	Digits	Iris	Segment	Vehicle	Waveform
Majority	0.60	4.65	10.85	42.70	20.83
Borda	2.60	5.43	10.85	38.72	24.65
Sum rule	0.30	4.65	6.02	33.05	18.28
Product rule	5.60	5.43	6.47	33.05	17.45
min rule	5.60	5.43	9.76	45.39	26.39
max rule	2.50	5.43	7.91	40.99	20.52
median rule	0.30	4.65	8.26	36.17	19.34
wBorda	0.20	4.65	9.11	35.60	18.88
trimmed means	0.30	5.43	6.12	32.91	19.03
spread	2.4	4.65	7.02	35.60	18.01
Random1	3.7	6.20	13.49	43.12	27.92
Random2	1.2	6.20	8.01	36.74	22.91

Table 4
Initial Results from Simulated Classifiers (Error Rate)

	At u=90%			At u=75%			at u=50%		
	Identical	Near	Disparate	Identical	Near	Disparate	Identical	Near	Disparate
Majority	0.86	0.85	0.89	10.3	10.4	10.4	50.1	49.9	50.3
Sum rule	0.06	0.049	0.048	1.04	1.01	0.98	5.48	5.43	5.40
Product	0.01	0.01	0.007	0.18	0.175	0.172	1.52	1.47	1.48
Median	0.072	0.065	0.057	1.08	1.07	1.03	7.94	7.83	7.80
wBorda	0.022	0.024	0.02	0.37	0.37	0.37	3.55	3.48	3.48
Trimmed Means	0.049	0.048	0.049	1.03	1.02	1.00	4.41	4.34	4.38
Spread	2.87	2.64	1.80	9.35	9.08	8.49	22.9	22.7	22.8
Random1	9.79	9.26	8.55	23.6	23.4	22.8	45.2	45.0	44.9

All composite classifiers were compared against each other, for all recognition rates, and for various relationships between classifiers; for classifiers all having identical recognition rates, for classifiers within 10% of each other (*near*), and within 20% (*disparate*). For this simulation, the error rate was collected and used as the basis for the comparison. The results are given in Table 4 for those combination methods that performed well on the real data sets. The first set of three columns show error rates for classifiers providing at least 90% recognition, which is followed by columns for 75% and 50% respectively.

The simulated classifiers used in this test generate independent classifications, which may partly explain the success of the product rule on these data. As a final test, another set of simulated data was generated that contained five classifiers, three of which were correlated to the 0.95 level. The results of the combination algorithms on these data was significantly different, as might have been predicted, and is summarized in table 5. In this case, all classifiers had recognition rates within 10% of each other (*near*).

5. Conclusions

In Table 2 the error rates for single classifiers give a minimum acceptable error for each data set; the combined error rate, using whatever combination rule we choose, must be better than the best single error rate. The product rule and the min rule fail this test on *digits*; all fail on *iris* (but a few are as good as the best single classifier); only spread, trimmed means, the product rule and the sum rule pass for the *segment* data set; majority vote, Borda, min and max rules, and random1 fail on *vehicle* data; and the same methods plus *random2* fail on waveform data. Only the sum rule was as good or better than all single classifiers on all of the real data sets. The original data from the classifiers was simple Type I classifications, and the response vectors on which the sum rule was based came from the confusion matrices for the simple classifiers. It appears that this scheme is useful in practice on real data.

The results from the simulated data are more difficult to summarize. Table 4 shows a sample of the results obtained by varying the specified recognition rates on the simulated classifiers. While not 'real' data, these represent more controlled experiments than are possible with actual classifiers. All of the combination methods except for Random1 perform better than the majority vote, again demonstrating the advantage of using response vectors based on the confusion matrix. A small surprise was the performance of wBorda, which deserves further examination. The degree to which the response-based methods were superior to the majority vote indicates that the small extra effort involved was more than justified.

As was mentioned previously, the simulated classifications are essentially independent. There are many ways in which classifiers can reinforce or conflict with each other, and these depend on the degree of correlation between them. As only one example, table 5 shows the results from five classifiers when three of them are correlated at the

95% level. Note that the performance of the product rule, which depends more than the others on independence, drops off significantly, while some of the other classifiers retain low error rates. In particular, the trimmed means, sum rule, and wBorda combiners perform well over all of the data in tables 3, 4 and 5, and all of the methods outperform the majority vote on correlated classifications. A detail not shown in table 5 is that the trimmed mean combiner has the lowest error rate from average individual rates between 95%-82% and below 56%, with wBorda having the lowest rates between 56% and 81%. Of course, the point was to demonstrate the effectiveness of creating responses from simple classifications, not to determine the best overall combination method.

The use of simulated classifiers requires further energy and study. The degree to which simulation can be used to evaluate composite classifiers depends on the degree to which the simulated classifiers can be shown to be representative of the real thing. The degree of control available to the experimenter using the simulation makes it an extremely attractive solution.

6. Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

7. References

1. Baraghimian, G.A., Klinger, A., "Preference Voting for Sensor Fusion", *SPIE Sensor Fusion III*, Orlando, Fl. April 19-20, 1990.

2. Jean-Charles de Borda, "Memoire sur les Elections au Scrutin", *Histoire de l'Academie Royale des Sciences*, Paris, 1781.

3. Marquis de Condorcet, "Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix" (Essay on the Application of Analysis to the Probability of Majority Decisions), Paris, 1785.

4. Wade D. Cook and M. Kress, "Ordinal Information & Preference Structures", Prentice Hall, Englewood Cliffs, N.J. 1992.

5. B. V. Dasarathy, "Asymmetric Fusion Strategies for Target Detection in Multisensor Environments", Proceedings of the SPIE, Vol. 3067, Sensor Fusion: Architecture, Algorithms, and Applications, pp. 26-37, April 1997.

6. P. Fishburn, "Preference Structures and Their Numerical Representations", ORDAL'96, Ottawa, Aug 5-9, 1996.

7. T.K. Ho, J.J. Hull, and S.N. Srihari, "Decision Combination in Multiple Classifier Systems", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 1, 1994. Pp. 66-75.

8. R.C. Holte, "Very Simple Classification Rules Perform Well On Most Commonly Used Data Sets", Machine Learning, Vol. 11, 1993. Pp. 63-90.

9. F. Kimura and M. Shridhar, "Handwritten Numeral Recognition Based on Multiple Algorithms", *Pattern Recognition*, Vol. 24, No. 10, 1991. pp 969-983.

10. J. Kittler, M. Hatef, and R.P. Duin, "Combining Classifiers", Proceedings of the ICPR, 1996. pp 897-901.

11. J.R. Parker, "Voting Methods for Multiple Autonomous Agents", Proc. ANZIIS'96, Perth, Australia, 1996.

Table 5: Initial Results from Simulated Correlated Classifiers (Error Rate)

	At u=90%	At u=75%	At u=50%
Majority vote	1.30	11.9	50.13
Sum rule	0.23	3.14	17.77
Product	0.40	3.45	25.30
Median	0.24	3.14	17.58
wBorda	0.27	2.75	16.06
Trimmed Means	0.22	2.98	15.2
Spread	6.01	17.10	38.52

12. J.R. Parker, “Evaluating Classifier Combination Using Simulated Classifiers”, University of Calgary Department of Computer Science Research Report #2000/659/11

13. Quinlan, J.R., “C4.5: Program for Machine Learning”, Morgan Kaufmann, San Francisco. 1993.

14. P.D. Straffin, Jr., “Topics in the Theory of Voting”, Birkhauser, Boston, 1980.

15. K. Tumer and J. Ghosh, “Classifier Combining through Trimmed Means and Order Statistics”, Proceedings of the international Joint Conference on Neural Networks, pp. 757-762, May 1998, Anchorage, AL.

16. K. Tumer and J. Ghosh, “Order Statistics Combiners for Neural Classifiers," Proceedings of the World Congress on Neural Networks, pp. I:31-34, July 1995, Washington, DC.

17. I.H. Witten and E. Frank, “Practical Machine Learning: Tools and Techniques with Java Implementations”, Morgan Kaufmann, San Francisco. 2000.

18. D.H. Wolpert, “Stacked Generalization”, Neural Networks, Vol. 5, 1992. Pp. 241-259.

19. L. Xu, A. Krzyzak, and C.Y. Suen, “Methods of Combining Multiple Classifiers and their Applications to Handwriting Recognition”, *IEEE Trans. SMC*, vol. 22, No. 3, 1992. pp 418-435.