

A Method to Calculate Classifier Correlation for Decision Fusion

Kai Goebel, Weizhong Yan, and William Cheetham

Abstract-- This paper deals with the concept of correlation in multi-classifier fusion tasks. The need to encapsulate the degree of correlation stems from the notion that the least amount of correlation among the classifiers is desired in fusion tasks. We define a measure that captures the correlation for n classifiers for binary output. In addition, we propose correlation calculation for classifiers with continuous output. We then suggest their use in classifier selection, simulation, and within the fusion task. Using an application from fault detection for uninhabited autonomous vehicles we show how the concepts can be applied.

Index Terms—Classification, Correlation, Information Fusion, Redundancy.

I. INTRODUCTION

CLASSIFIER fusion attempts to aggregate the output from several classification tools as well as secondary information where available. This is mandated by the limited performance of any single classifier, among other things. An issue that has not been addressed in depth is the need to use tools that – besides offering reasonable performance – are correlated only to a certain extent. This is necessary to avoid the problems introduced though aggregation schemes that are biased by redundant information. If, for example, two classifiers in a three classifier fusion task are completely redundant, many fusion schemes are thrown off by the repetitive information. While some degree of confirmatory information is desired, it is the complementary information that allows the multi-classifier fusion to be successful. The paper will first discuss classifier fusion in section II, then address correlation analysis in section III, followed by application to a fault detection example in section IV

II. CLASSIFIER FUSION

To accommodate particular classification requirements, different classification tools are sometimes developed either in parallel or sequentially resulting from a need for increased

class coverage or for better classification performance. The stipulation to use several tools arises in part because often times any one tool cannot deal with all classes of interest at the desired level of accuracy. Other reasons include the historic use of particular tools that do not have expansion capabilities, limited resources to develop a comprehensive classification tool addressing all issues, etc. In addition, tools' classification capabilities are impacted differently by environmental changes. While the resulting patchwork approach might achieve optimization at particular local level, it might also blur the picture for other existing classifiers due to conflicting information. There is a potential benefit gained by taking a system-level view. Such a system-level scheme gathers and combines the results of different classification tools to maximize the advantages of each one while at the same time minimizing the disadvantages. This fusion scheme holds the promise to deliver a result that is better than the best result possible by any one tool employed. In part this can be accomplished because redundant information is available, which when combined correctly improves the estimate of the better tool and compensates for the shortcomings of the less capable tool. However, there is no substitute for a good classification tool and, ordinarily, multiple, marginal-performance tools do not necessarily combine to produce an improved result and in fact may worsen the outcome [1].

Classifier performance is problem dependent. The decisive selection criteria are classification performance and execution time. Another important fact considered during the classifier selection process is the correlation between the classifiers selected. The correlation between the classifiers to be fused needs to be small to enable performance improvement [2, 3] in classifier fusion.

A. Classifier performance evaluation

We consider classifier problems where a feature vector $x \in \mathcal{R}^p$ is to be labeled into one or more of c classes. In order to achieve high overall performance of the fault detection function, the performance of each individual classifier has to be optimized prior to using it within any fusion schemes. The optimization is performed with respect to selecting appropriate parameters and – where applicable – structure that govern the performance.

For each classifier, a confusion matrix M can be generated using the labeled training data [4]. The confusion matrix lists the true classes c versus the estimated classes \hat{c} . Because all

Manuscript received June 29, 2001. This work was supported in part by the Knowledge Management MGPD and by the GE-Lockheed Martin Shared Vision Program.

K. Goebel is with GE Corporate Research & Development, K1-5C4A, One Research Circle, Niskayuna, NY 12309 USA (telephone: 518-387-4194, e-mail: goebelk@crd.ge.com).

W. Yan is with GE Corporate Research & Development, K1-5B34B, One Research Circle, Niskayuna, NY 12309 USA (telephone: 518-387-5704, e-mail: yan@crd.ge.com).

W. Cheetham is with GE Corporate Research & Development, K1-5C21A, One Research Circle, Niskayuna, NY 12309 USA (telephone: 518-387-5222, e-mail: cheetham@crd.ge.com).

classes are enumerated, it is possible to obtain information not only about the correctly classified states (N^{00} and N^{11}), but also about the false positives (N^{01}) and false negatives (N^{10}). The top-left entry of the confusion matrix is dedicated to the normal case N^{00} . The first row – except the first entry – contains the N^{01} . The off-diagonal elements – except the first row – contains the N^{10} . Sometimes a further distinction is made between false negatives and false classifieds where the false classifieds are defined to be the off-diagonal elements of the confusion matrix except the first row and the first column. A typical two-class confusion matrix M is shown in Fig. 1.

		Classes assigned by Classifier	
		0	1
True Classes	0	N^{00}	N^{01}
	1	N^{10}	N^{11}

Fig. 1: Typical 2-class confusion matrix

From the confusion matrix of each classifier, the false positive (FP) error, the false negative (FN) error, the total error rate (TER), and the total success rate (TSR) can be calculated for the classifier. All of these error rates are defined as in Equations 1 – 4. The total error rate (TER) or the total success rate (TSR) is typically used as a simple measure for overall performance of a classifier:

$$FP = \frac{N^{01}}{N^{00} + N^{01}} \quad (1)$$

$$FN = \frac{N^{10}}{N^{10} + N^{11}} \quad (2)$$

$$TER = \frac{N^{01} + N^{10}}{N^{00} + N^{11} + N^{01} + N^{10}} \quad (3)$$

$$TSR = 1 - TER \quad (4)$$

III. CLASSIFIER CORRELATION ANALYSIS

While it has been well understood that each individual classifier's performance is very important to the performance of a classifier fusion there seems to be less awareness that the dependency between the classifiers to be fused also affects the fusion results. Studies [3] have shown that the degree of correlation between the classifiers adversely affects the performance of the subsequent classifier fusion. If two classifiers agree everywhere, the fusion of the two classifiers will not achieve any accuracy improvement no matter what fusion method is used. For classifier fusion design, classifier correlation analysis is, therefore, equally important as the classifier performance analysis.

1) Two class correlation analysis

Petrakos et al. [3] describe a classifier correlation analysis for two classifiers. Based on the classifier outputs on the labeled training data, a 2x2 matrix N as shown in Fig. 2 can be generated for each classifier pair. The off-diagonal numbers directly indicate the correlation degree of the two classifiers. The smaller the two off-diagonal numbers are, the higher the correlation between the two classifiers will be. The proportion of specific agreement which we call here the correlation index, ρ_2 , is defined as [3]

$$\rho_2 = \frac{2 \times N^{FF}}{N^{TF} + N^{FT} + 2 \times N^{FF}} \quad (5)$$

where N^{TT} implies that both classifiers classified correctly, N^{FF} means both classifiers classified incorrectly, N^{TF} represents the case of the 1st classifier classified correctly and 2nd classifier classified incorrectly, and N^{FT} stands for the 2nd classifier classified correctly and 1st classifier classified incorrectly as further shown in Fig. 2. In order for classifier fusion to be effective in performance improvement, the correlation index, ρ_2 , has to be small (low correlation).

		Classifier # 2	
		T	F
Classifier # 1	T	N^{TT}	N^{TF}
	F	N^{FT}	N^{FF}

Fig. 2: Correlation analysis matrix

Consider a 2 tools situation and associated experiment results as shown in Table 1:

Table 1: Results from experiment for 2 classifiers

Answer tool 1	Answer tool 2
T	T
T	F
F	T
T	F
F	F
F	F
T	F
F	T
T	T
T	T
T	T
T	T
T	T
T	F
T	T
T	T
F	T

The calculation of ρ_2 yields $\rho_2 = 0.36$.

Had tool 2 been completely redundant to tool 1, the correlation index would have been $\rho_2 = 1$

2) n Class correlation analysis

We propose an extension of the 2 class correlation coefficient to n different classifiers. The notion that redundancy is described by the individual true and false answers of the classifiers is retained from the 2 class correlation analysis. The larger the correlation index, the larger the redundancy. In particular, the correlation index goes to zero if the individual incorrect answers are disjoint for all answers. In other words there is always at least one correct answer for any class. The correlation coefficient gets larger as the number of wrong answers are the same for many answers. Let N^f be the number of experiments where all tools had a wrong answer, N_i^c be the number of experiments with combinations of correct and incorrect answers; c is the combination of correct and incorrect answers (for 2 tools: $c \in \{wr, rw\}$; for 3 tools: $c \in \{wwr, wrw, rww, wrr, rwr, rrw\}$, etc.); n is the number of tools. The correlation coefficient is

$$\rho_n = \frac{nN^f}{\sum_{i=1}^{2^n-2} N_i^c + nN^f} \quad (6)$$

If N is the number of experiments and N^f is the number of experiments for which all tools had a right answer to, equation 6 can more conveniently be rewritten by:

$$\rho_n = \frac{nN^f}{N - N^f - N^t + nN^f} \quad (7)$$

Consider a 3 tool scenario which is a repetition of the example for two tools with the addition of the third tool such that it gets the answer wrong in 50% of the cases. The calculation of ρ_n yields: $\rho_n = 0.21$

Although the newly added tool has poor performance, its addition reduces the overall redundancy of the tool assembly.

It is interesting to note that the correlation index does not record redundancy with any particular tool (for $n > 2$) but with a set of tool only. Consider the following two cases shown in Table 2 and Table 3:

Table 2: Output for 3 classifiers

Answer tool 1	Answer tool 2	Answer tool 3
T	F	F
F	T	F
F	T	T
T	T	T
F	F	F

The correlation index is $\rho_n = 0.5$

Table 3: Output for 3 classifiers with different output for 3rd tool

Answer tool 1	Answer tool 2	Answer tool 3
T	F	T
F	T	T
F	T	F
T	T	T
F	F	F

The correlation index is $\rho_n = 0.5$

Obviously the third tool is different in the two cases. However, the degree of correlation is the same because it does not matter whether it is correlated to the first or to the second tool. Rather it is only relevant that it correlated to the combination of the first two tools.

It has to be noted that the calculation of the correlation factor can be performed on multi-class scenarios as well because the factor is only concerned with the correctness of the outcome.

B. Classifiers with continuous output

For classifiers that give continuous output such as confidences, partial class membership etc. we propose a slightly different operator ρ_{n_c} . Let N_c^f be the sum of all false tool outputs that are greater than threshold t_c ; o_i^f is the number of aggregated

false output per case i , $o_i^f = \frac{\sum_{j=1}^{n_{tools}} o_{j,i}^f}{n_{tools}}$ and

$$o_{j,i} = \begin{cases} o_{j,i}^{raw} & \text{if } o_{j,i}^{raw} > t_c \\ 1 - o_{j,i}^{raw} & \text{otherwise} \end{cases}; t_c \text{ is the decision threshold for}$$

class membership ($t_c = 0.5$); N_j is the number of cases that are

$$\text{false per threshold } t. N_c^f = \sum_{i=1}^{N_j} o_i^f$$

Then the correlation index ρ_{n_c} is

$$\rho_{n_c} = \frac{nN_c^f}{N - N_c^f - N^t + nN_c^f} \quad (8)$$

Consider the output of 2 tools with continuous output bounded by [0,1] as shown in Table 4. Table 4 also shows the normalized adjusted output o .

Table 4: Tool output for 2 classifiers in continuous format

True state	Output tool 1	Output tool 2	Normalized adjusted cumulative output o_i
1	0.7	0.6	0.65
1	0.6	0.3	0.7
1	0.3	0.6	0.65
0	0.2	0.7	0.75
0	0.6	0.8	0.7
0	0.9	0.8	0.85
0	0.1	0.6	0.75
1	0.2	0.6	0.75
0	0.4	0.3	0.65
0	0.4	0.4	0.6
1	0.8	0.6	0.7
0	0.2	0.1	0.85
0	0.3	0.7	0.7
1	0.7	0.9	0.8
0	0.4	0.4	0.6
0	0.7	0.4	0.65

The calculation of ρ_{n_c} yields: $\rho_{n_c} = 0.2441$

where

$$N_c^f = 1.55$$

$$N_c^t = 4.85$$

IV. USE OF CORRELATION INDEX

The correlation coefficient can be used for different purposes, in particular for 1.) tool selection; 2.) tool simulation; and 3.) fusion estimate refinement

A. Tool selection

Tool selection should be carried out such that the least redundancy is maintained. Starting with the best performing tool using false positives, false negatives, and false classified states for evaluation, the next best tool will be added. This process is repeated until the desired number of classifiers has been reached or until the correlation index increases. Consider the following example:

Table 5: 2 class output

Answer tool 1	Answer tool 2
T	T
T	F
F	T
F	F
F	F

The correlation index is $\rho_n = 0.67$

Adding a third tool that is completely redundant to the second,

Table 6: 3rd tool added

Answer tool 1	Answer tool 2	Answer tool 3
T	T	T
T	F	F
F	T	T
F	F	F
F	F	F

The correlation index is $\rho_n = 0.75$, i.e., the correlation index increased. This tools should not be considered for fusion. Had the third tool instead been as shown in Table 7, the index would have been $\rho_n = 0.5$; i.e., the correlation index decreased.

Table 7: Different 3rd tool

Answer tool 1	Answer tool 2	Answer tool 3
T	T	T
T	F	F
F	T	T
F	F	T
F	F	F

B. Tool Simulation

Establishing the degree of correlation is also important for simulation purposes. When testing and validating fusion systems, it is important to take into account the degree of correlation. If systems are treated as quasi-independent, incorrect conclusions might be drawn about their performance. Therefore, a simulation must always consider the interdependence of classifiers. One way to accomplish that is to perform a guided Monte Carlo simulation. This guided Monte Carlo simulation is carried out such that output for the prime classifier is generated according to performance criteria as found in the confusion matrix. In particular, values are simulated with values greater than 0.5 for the percentage indicated in the diagonal entries of the confusion matrix and values less than 0.5 in the remaining cases. Appropriate distributions must be used such as monotonic increasing Gaussian [4] centered around 1 or Weibull distributions centered around 0.5.

The secondary classifier, that is, the classifier that is ranked second using some overall performance measure such as false positives and false negatives [4] must be simulated to account for the correlation established. This can be done by guiding the Monte Carlo simulation such that based on the value of the primary classifier, the value of the secondary classifier is modeled.

Next, the value of the tertiary classifier is modeled based on the combination of the values of the first two classifiers. The information used for that purpose is encoded in the truth tables amended by the correlation information. The

Consider the following example for two classifiers with truth table as shown in Table 8:

Table 8: Classifier output and cumulative correlation

Classifier 1	Classifier 2	Correlation
T	T	.65
T	F	.05
F	T	.1
F	F	.2

If a third classifier is added to the distribution above, the distribution of the first two classifiers stays the same. However, each row in the table for two classifiers is split into two rows. The sum of the correlation in the split rows is the same as the correlation in the original row. For three classifiers, the truth table might look as shown in Table 9:

Table 9: Classifier output and cumulative correlation for 3 tools.

Classifier 1	Classifier 2	Classifier 3	Correlation	Cum. Score
T	T	T	.35	.35
T	T	F	.30	.65
T	F	T	.02	.67
T	F	F	.03	.70
F	T	T	.05	.75
F	T	F	.05	.80
F	F	T	.05	.85
F	F	F	.15	1.0

The pseudo code for generating correlated values is as follows (using, without loss of generality, uniformly distributed output):

```

loop through number of cases:
  this_case ← find(rand ≤ truth_table(:, truth_table_size(1,2)));
  loop through number of classifiers
    if case to be simulated is true:
      case_value ← rand;
      if case_value > 0.5:
        retain value
      else
        case_value ← 1 - case_value
    else:
      case_value ← rand;
      if case_value < 0.5:
        retain case_value
      else:
        case_value ← 1 - case_value

```

C. Using correlation information in fusion tasks

One way integrate correlation information into the fusion task is by emphasizing weights for classes where the correlation is smaller. Obviously, the intuitive approach is to trust the best classifier for a given class most. But this does not take into consideration the correlation other than using the reliability of the tools in the scaling process. To accomplish an integration of the correlation, one can weight more heavily the less

reliable tool in observed class constellations where low correlation is observed. Conversely, where high constellation is observed, the less reliable tool should be weighted less heavily. For example, in the case of the first case study, the constellation {x, c2} (where x is don't care) is weighted more heavily for tool 3. The following pseudo code captures that notion.

```

for i ∈ number of data :
  old_fused_value ← 1
  for i ∈ number of tools :
    if tool_X(j) > 0:
      fused_value(i) ← old_fused_value ·
        data(i,1:j) · correlation(j)
      old_fused_value ← fused_value(i)
    end
  end
end

```

V. APPLICATION TO FAULT DETECTION IN UAV

A. UAV

Unlike a piloted aircraft where the pilot's skill and predictive capabilities ensure the safety of aircraft under adverse conditions, the Uninhabited Autonomous Vehicles (UAV) has to rely on the mission manager to achieve the safety level that is comparable to that of the piloted aircraft. To meet this safety goal, the mission manager and its control system must be capable of rigorously analyzing and predicting in real time component failures to determine appropriate response much as a pilot.

B. UAV Classifier selection

Three classifiers were selected for the fault detection of the UAV: 1.) binary decision tree [5]; 2.) fuzzy k-nearest neighbor [6], and 3.) multiple-layer feed-forward neural network. All of the three classifiers are non-parametric; however, the classifiers are based on different models. The decision tree classifier is rule-based, and the fuzzy k-nearest neighbor classifier is distance-based, while the neural network classifier is weight-based.

C. Experimental Results

1) Data Generation

For the initial prototyping design, the real-time air vehicle simulation and visualization software FlightMaster from Sight, Sound and Motion, Inc. (SSM) was used to generate data with normal engine condition under various flight conditions. We then repeated the runs under the same flight conditions, but with faults injected to the engine model. The simulation runs cover the flight operation space defined by the altitude ranging from 10k to 30k and the Mach number from 0.4 to 0.9. Because the training data obtained from running the SSM simulation are noise-free we add a uniformly distributed noise

with a magnitude of 2 times of the standard deviation to validate the robustness of the fault detection function.

The three features selected for detecting the engine system faults are the fuel flow rate, the engine percentage power, and the engine RPM. The residuals for the normal engine are generally scattered around zero while the fault engine residual are scattered further away from the zero point. There is a certain amount of overlap between the two residuals (normal and fault). The residual overlapping is the major source of the classification errors of all classifiers (Table 10).

Table 10: Confusion matrices for UAV classifiers

		Normal	Fault
fuzzy knn	Normal	147	48
	Fault	3	102
C4.5	Normal	138	34
	Fault	12	116
NN	Normal	143	35
	Fault	7	115

Table 11 lists the cumulative true and false assignments for the different tool combinations. Note that combination 1-2-3 has the lowest overall false assignment. Although combinations 1-3 and 2-3 have the same cumulative false assignment, the correlation coefficient for combination 2-3 is lower because the cumulative true assignment is higher. The result justifies the use of the three classifiers.

Table 11: Correlation coefficient for different tool combinations

	N^f	N^t	ρ
Tool 1	104	196	1
Tool 2	128	172	1
Tool 3	122	178	1
Combination Tools 1-2	36	238	0.7347
Combination Tools 1-3	32	244	0.766
Combination Tools 2-3	32	244	0.7273
Combination Tools 1-2-3	29	235	0.7073

Table 12 shows that the weighted majority vote scheme shows identical classification performance (TSR) as the best individual classifier (the neural network classifier). Of the four classifier fusion schemes, the Naïve Bayes scheme as well as the Correlation based fusion scheme achieve classification performance that is better than that of the best individual classifier with a performance improvement margin of slightly over 1%. Given the sparse data sets, there are obvious limits to the potential improvement for any fusion scheme.

Table 12: Results of classifiers and classifier fusion schemes

	FPE (%)	FNE (%)	TER (%)	TSR (%)
Fuzzy k-NN	2.0	32.0	17.0	83.0
Decision Tree	8.0	22.7	15.0	85.0
Neural Network	4.6	23.3	14.0	86.0
Majority vote (MV)	3.3	27.3	15.3	84.7
Weighted MV	4.6	23.3	14.0	86.0
Naïve-Bayes	4.6	23.3	13.0	87.0
Correlation based	4.6	23.3	13.0	87.0

CONCLUSIONS

This paper introduced a correlation coefficient that measures the correlation between n classes for crisp as well as for soft class assignment. The correlation coefficient can be used for classifier selection, classifier simulation, and fusion. The properties of the correlation coefficient are further demonstrated on a small data set serving as an illustrative example only which was motivated from an UAV fault detection application. Although the correlation was the only parameter considered in the fusion task, it is envisioned that knowledge about the degree of correlation is only one of many parameters that will be considered in a more comprehensive fusion scheme. Fundamentally, a poorer performing tool may have the potential to help the fusion scheme as shown in the illustrative example above. If several subsets of tools have equal correlation, the choice to which tools to use can be made based on which faults the tools missed on. This choice can be made on sensitivity to false positives versus false negatives. It can also be made on the criticality of certain faults because often times, not all faults are equally important.

REFERENCES

- [1] D. Hall, A. Garga. "Pitfalls in Data Fusion (and How to Avoid Them)". *Proc. Second Int. Conf. Information Fusion (Fusion '99)*, pp. 429-436, 1999.
- [2] T. Ho, J. Hull, and S. Srihari, "Decision Combination in Multiple Classifier Systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No.1, 1/1994.
- [3] M. Petrakos, I. Kannelopoulos, J. Benediktsson, and M. Pesaresi, "The Effect of Correlation on the Accuracy of the Combined Classifier in Decision Level Fusion", *Proceedings of IEEE 2000 International Geo-science and Remote Sensing Symposium*, Vol. 6, 7/2000.
- [4] K. Goebel, "Architecture and Design of a Diagnostic Information Fusion Tool", *AIEDAM: special edition AI in Equipment Service*, vol. 15 no. 4, pp. 335-348, 2001.
- [5] J. Quinlan, *C4.5--Programs for machine learning*, Morgan Kaufmann, San Mateo, CA, 1993.
- [6] J. Keller, M. Gray, and J. Givens, Jr., "A Fuzzy K-Nearest Neighbor Algorithm", *IEEE Transactions on Systems, Man, Cybernetics*, Vol. SMC-15, No.4, 1985.