An Algorithm for the Optimization of Multiple Classifiers in Data Mining Based on Graphs

Andrei Kelarev[1], Joe Ryan[2], John Yearwood[1]
[1]School of ITMS, University of Ballarat,
P.O. Box 663, Ballarat, Victoria 3353, Australia
{a.kelarev,j.yearwood}@ballarat.edu.au
[2]School of Electrical Engineering and
Computer Science, University of Newcastle,
Callaghan, NSW 2308, Australia
Joe.Ryan@newcastle.edu.au

Abstract

This article develops an efficient combinatorial algorithm based on labeled directed graphs and motivated by applications in data mining for designing multiple classifiers. Our method originates from the standard approach described in [37]. It defines a representation of a multiclass classifier in terms of several binary classifiers. We are using labeled graphs to introduce additional structure on the classifier. Representations of this sort are known to have serious advantages. An important property of these representations is their ability to correct errors of individual binary classifiers and produce correct combined output. For every representation like this we develop a combinatorial algorithm with quadratic running time to compute the largest number of errors of individual binary classifiers which can be corrected by the combined multiple classifier. In addition, we consider the question of optimizing the classifiers of this type and find all optimal representations for these multiple classifiers.

1 Introduction

This paper uses labeled directed graphs to define multiple classifiers and investigate their properties. Graph labelings have received serious attention in the literature and a lot of interesting results have been obtained (see,

JCMCC 71 (2009), pp. 65-85

for example, [5, 9, 15, 24, 26, 33, 36]). For an overview of relations of this direction to automata theory the readers are referred to [20, 21, 22, 24, 25, 27, 29, 30].

A well known method of designing efficient multiple classifiers consists in representing them as several binary classifiers combined in one scheme. This method is very effective, and it is often advisable to apply it even in situations where it is possible to build multiclass classifiers analyzing the data directly, see [37], Section 7.5. Classification of data plays one of the central roles in data mining and in practical applications of artificial intelligence methods in general (see, in particular, [38] and [7, 8, 18, 23, 28, 31]).

The main advantage of using combined multiple classifiers is that they can correct errors of individual binary classifiers and produce correct classifications despite individual classification errors. It is usually desirable to choose convenient representation for the class set of the multiple classifier and to ensure that it has a small set of generators.

The problem of finding the number of errors of individual binary classifiers that a multiple classifier can correct in general is rather complicated. It is well known that in full generality this problem is related to several other very difficult algorithmic problems, see [12, 16, 17, 19, 35, 38].

It is remarkable that in our situation it has turned out possible to reduce this problem and solve it with a fairly simple algorithm based on directed graphs. The reduction however is rather nontrivial, and the most complicated part of our paper is devoted to proving that it indeed achieves the objective, and that our simple and efficient algorithm indeed accomplishes the task.

2 Preliminaries and technical definitions

We use standard notation and terminology, following [14, 19, 20, 31, 35, 37, 38]. Let $D = (V, A)$ be a directed graph with the set

$$V = \{v_1, \ldots, v_n\}$$

of vertices and a set

$$A = \{a_1, \ldots, a_m\}$$

of arcs without multiple arcs but possibly with loops. All graphs considered in this paper will be subgraphs of D, and so will possess all the inherited properties.

We are going to use the set A of arcs to represent multiple classifiers. The number of arcs will be equal to the number of binary classifiers being combined. Denote all the binary classifiers by $b_1, \ldots, b_m$, and suppose that the output of each of them is either 0 or 1. If $o_1, \ldots, o_m$ are the outputs of the binary classifiers, then the sequence $(o_1, \ldots, o_m)$ is called a *class vector* of the combined multiple classifier, and the set of all class vectors is called the *class set.* Each class vector represents one class in the classification produced by the multiple classifier, see [37], Table 7.1.

Let $F = GF(2)$ be the finite field of order two, i.e., the set $\{0, 1\}$ with the standard addition and multiplication. Let us denote by F^m the set of all sequences of all combinations of outcomes of the binary classifiers which can occur in general. This means that

$$F^m = \{(r_1, \ldots, r_m) \mid r_1, \ldots, r_m \in F\}. \tag{1}$$

For every element $r \in F^m$, denote by r_i the i-th component of the sequence r, so that

$$r = (r_1, \ldots, r_m) \in F^m. \tag{2}$$

In order to emphasize the relation of elements of the set to the arcs of the graph D we will also use an alternative shorter symbolic notation. Obviously, each arc $a_i \in A \subseteq F^m$, for $1 \leq i \leq m$, corresponds to the sequence with i-th coordinate equal to 1 and all other coordinates zero. Therefore, we can rewrite (2) and see that each sequence $r = (r_1, \ldots, r_m)$, where $r_1, \ldots, r_m \in F$, can be also recorded as the sum of arcs with their respective coefficients as follows

$$\begin{aligned} (r_1, \ldots, r_m) &= \sum_{i=1}^{m} r_i a_i \\ &= \sum_{i=1}^{m} r_i (u_i, v_i), \end{aligned} \tag{3}$$

where $a_i = (u_i, v_i) \in A$. This notation is convenient for defining particular sequences by indicating which elements in the sequence are nonzero, and also when one has to see which vertices are involved, which is often useful in proofs.

If we ignore the ordering of arcs used in the sum (3), then clearly every element r of F^m has a unique representation as a sum of the form

$$r = \sum_{(u,v) \in A} r_{(u,v)}(u, v) = \sum_{e \in A} r_e e, \tag{4}$$

where $r_e = r_{(u,v)} \in F$. Comparing (3) and (4), we see that if

$$A = \{a_1, \ldots, a_m\},$$

then

$$r_{a_i} = r_i \text{ for } i = 1, \ldots, m. \tag{5}$$

In order to generate classifiers with known properties and find optimal multiple classification schemes, the set V of vertices and relations among arcs are used to introduce additional structure for the classifier. The structure enables one to find small generating sets for the classifier. Let us first explain the basic essential properties required of the class sets.

The *Hamming weight* or *weight* $\mathrm{wt_H}(c)$ of a sequence c in F^m is the number of nonzero coordinates in c. The *weight* of a class set C is the minimum weight of a nonzero element in C. The *minimum distance* of a class set C is the minimum number among all weights of nonzero differences between pairs of elements in C. Evidently, if a class set forms a linear space, then its minimum distance is equal to its weight. It is well-known and easy to verify that the number of errors of binary classifiers, which the multiple classifier can correct, is equal to $\lfloor (d-1)/2 \rfloor$, where d is the minimum distance of the class set.

The *information rate* of a class set C in F^m can be defined as $\log_2(|C|)/m$. It shows how large is the proportion of output of the binary classifiers used to produce the outcomes of the multiple classification as opposed to some additional efforts spent on increasing reliability and correcting classification errors.

All sequences of the class set C can be written down in a matrix M to discuss their properties. If M has two identical columns, this means that two binary classifiers produce identical outputs. Although that classifier could help to correct classification errors, this duplication is very inefficient. Therefore in a situation like this one of these classifiers can be removed and a better scheme can be devised. Likewise, it is undesirable to have strong correlation or functional dependencies between very small sets of columns in M or between binary classifiers.

Thus, the class set C defining a multiple classifier must satisfy the following most essential basic properties:

(1) The Hamming weight of C must be large.

(2) The information rate of C must be large.

(3) A convenient method of generating the set C is essential.

(4) The matrix representation of C should not have duplicate columns.

(5) If all vectors of C are recorded in a matrix, then there should not be strong correlation or functional dependencies between small sets of columns of the matrix.

Additional properties may also be required and have to be investigated depending on the practical task being considered.

Instead of storing the whole large class set C in computer memory, it is convenient to be able to generate C with one or more generators. To this end we introduce a multiplication on the set F^m. It allows us to multiply the generators with arbitrary elements of F^m and to take their sums. Recall, that as usual the standard addition is defined on F^m componentwise, i.e., the sum of two arbitrary sequences $(r_1, \ldots, r_m)$ and $(s_1, \ldots, s_m)$ in F^m is defined as

$$\begin{aligned}(r_1, \ldots, r_m) + (s_1, \ldots, s_m) = \\ = \quad (r_1 + s_1, \ldots, r_m + s_m).\end{aligned} \tag{6}$$

An additional operation defined on the class set comes from the directed graph. Namely, the following rule defines formal products of sequences in the class set. It is easy to write this rule down in the case of sequences with only one nonzero coordinate corresponding to only one arc. Using the notation (4), let us take two arbitrary sequences of this kind, say (u_1, v_1) and (u_2, v_2). Then the rule is written down as

$$(u_1, v_1) \cdot (u_2, v_2) = \begin{cases} (u_1, v_2) & \text{if } v_1 = u_2, (u_1, v_2) \in A, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

see, for example, [19], §3.15. Accordingly, for arbitrary sequences

$$(r_1, \ldots, r_m) \quad = \quad \sum_{i=1}^{m} r_i(u_i, v_i), \tag{8}$$

$$(s_1, \ldots, s_m) \quad = \quad \sum_{i=1}^{m} s_i(u_i, v_i), \tag{9}$$

recorded in the form (3), where all r_i, s_i are in F, we have

$$\begin{aligned}\sum_{i=1}^{m} r_i(u_i, v_i) \cdot \sum_{j=1}^{m} s_j(u_j, v_j) = \\ = \quad \sum_{i=1}^{m} \sum_{u_j = v_i} r_i s_j(u_i, v_j).\end{aligned} \tag{10}$$

When it is essential to emphasize that the set F^m has been equipped with operations one of which is defined using a directed graph D, we will write

$$F^m = F^m_D.$$

Now we can use these two operations to generate classifiers. An element $r \in F^m$ is said to be *generated* by the elements $g_1, \ldots, g_k$ if it is the sum of multiples of these generators. Every generator is allowed to contribute several summands to this sum, and one has to write their multipliers on both sides since the multiplication does not commute. This means that r is generated by the elements $g_1, \ldots, g_k$ if and only if there exist $f_{j,i}, h_{j,i} \in F^m \cup \{1\}$ such that

$$r = \sum_{j=1}^{m_1} f_{j,1} g_1 h_{j,1} + \cdots + \sum_{j=1}^{m_k} f_{j,k} g_k h_{j,k}. \tag{11}$$

Each of the k sums in this expression can be rewritten as follows

$$\begin{aligned} \sum_{j=1}^{m_i} f_{j,i} g_i h_{j,i} &= \sum_{j=1}^{m_i} \left(\sum_{a=1}^{m} f_a^{(j,i)} \right) g_i \left(\sum_{b=1}^{m} h_b^{(j,i)} \right) \\ &= \sum_{j=1}^{m_i} \sum_{a=1}^{m} \sum_{b=1}^{m} \left(f_a^{(j,i)} g_i h_b^{(j,i)} \right). \end{aligned}$$

This is an expression of the same form as (11). Hence it follows that instead of multiplying the generator g_i by a whole sum of several arcs, we can multiply it by each arc separately, and then form the sum. Therefore, to simplify further notation it is always possible to assume that even the original multipliers $f_{j,i}$ and $h_{j,i}$ have this form from the very beginning, i.e.,

$$f_{j,i}, h_{j,i} \in \{r(u,v) \mid r \in F, (u,v) \in A\} \cup \{1\}. \tag{12}$$

Accordingly, a whole class set C of a multiple classifier is said to be *generated* by the elements $g_1, \ldots, g_k$ in F^m if C coincides with the set of all sums of multiples of these generators, i.e.,

$$\begin{aligned} C &= C(g_1, \ldots, g_k) \\ &= \left\{ \sum_{j=1}^{m_1} f_{j,1} g_1 h_{j,1} + \cdots + \sum_{j=1}^{m_k} f_{j,k} g_k h_{j,k} \;\middle|\; \text{where all } f_{j,i}, h_{j,i} \in F^m \cup \{1\} \right\}. \end{aligned} \tag{13}$$

In this case the notation $C = C(g_1, \ldots, g_k)$ is used when it is necessary to indicate the generators explicitly. Again we may assume that condition (12) holds, as we did in the case of expression (11).

Considering several generators will make it possible to generate class sets with larger information rates. Our objective is to optimize properties of these class sets and, in particular, to minimize the number of generators.

3 Main results

It is customary in the literature to assume that the graph D satisfies an additional property which is general enough and ensures that the operation introduced in (10) satisfies convenient properties and behaves like the familiar products of integers or polynomials, see [19]. Further we assume that the graph D satisfies the following property. A graph D is said to be *transitive*, if $(u,v),(v,w) \in A$ imply $(u,w) \in A$, for all $u,v,w \in V$. Graphs of this sort are very well known. This class is quite general and has been investigated in the literature (see, for example, [32]). In particular, there are several well-known algorithms for testing whether a given graph is transitive, for example, with running times $O(|V|^3)$ and $O(|V|(|V|+|A|))$ (see [14], § 12.4.2). For a transitive graph, it is straightforward to verify that (6) and (10) satisfy the same laws as the familiar operations on integers, polynomials or matrices. These laws provide flexibility and make the operations convenient.

It is well known that many problems concerning class sets of multiple classifiers are NP-complete. Therefore it is nice that, using labeled directed graphs, we have been able to develop a very efficient quadratic algorithm answering a natural question concerning class sets of multiple classifiers. Let us first include the pseudocode of our algorithm.

Algorithm 1 *Given a transitive directed graph $D = (V,A)$, returns a multiple classifier $C(h)$ in F_D^m capable of correcting the largest number of errors among all classifiers which can be generated by several elements in F_D^m and finds the error-correcting capability of $C(h)$.*

```
 1. int a = b = c = 0;
 2. for ( j = 1; j <= n; j++ )
 3.     { L(j) = R(j) = L'(j) = R'(j) = ∅; }
 4. for ( j = 1; j <= n; j++ )
 5.     for ( ℓ = 1; ℓ <= n; ℓ++ )
 6.         if ( (v_j, v_ℓ) ∈ A ) {
 7.             L(ℓ) = L(ℓ) ∪ {j};
 8.             R(j) = R(j) ∪ {ℓ};
 9.         }
10. for ( j = 1; j <= n; j++ )
```

```
11.     for ( ℓ = 1; j <= n; j++ )
12.         if ( (v_j, v_ℓ) ∈ A ) {
13.             if ( |L(j)|==0 )
14.                 L'(ℓ) = L'(ℓ) ∪ {j};
15.             if ( |R(ℓ)|==0 )
16.                 R'(j) = R'(j) ∪ {ℓ};
17.         }
18. Find a = max_j |R'(j)| and j_a with a = |R'(j_a)|;
19. Find b = max_ℓ |L'(ℓ)| and ℓ_b with b = |L'(ℓ_b)|;
20. Take w_c = 0 in F^m;
21. for ( j = 1; j <= n; j++ )
22.     for ( ℓ = 1; ℓ <= n; ℓ++ )
23.         if (|L(j)|==0 && |R(ℓ)|==0)
24.             { w_c = w_c + (v_j, v_ℓ); c++; }
25. if ( a == b == c == 0 )
26.     return 0;
27. else if ( a == max{a,b,c} ) {
28.     w_a = Σ_{ℓ∈R'(j_a)} (v_{j_a}, v_ℓ);
29.     return C(w_a), ⌊(a − 1)/2⌋;
30. }
31. else if ( b == max{a,b,c} ) {
32.     w_b = Σ_{j∈L'(ℓ_b)} (v_j, v_{ℓ_b});
33.     return C(w_b), ⌊(b − 1)/2⌋;
34. }
35. else
36.     return C(w_c), ⌊(c − 1)/2⌋;
```

Next we summarize the formal definitions of sets used in Algorithm 1. For $1 \leq i, \ell \leq n$, it finds the sets

$$L(\ell) = \{j \mid (v_j, v_\ell) \in A\}, \tag{14}$$

$$R(j) = \{\ell \mid (v_j, v_\ell) \in A\}, \tag{15}$$

and then finds all the sets

$$L'(\ell) = \{j \in L(\ell) \mid L(j) = \emptyset\}, \tag{16}$$

$$R'(j) = \{\ell \in R(j) \mid R(\ell) = \emptyset\}, \tag{17}$$

Then it computes the maximum values and finds the first integers j_a, ℓ_b such that

$$a = |R'(j_a)| = \max_j |R'(j)| \tag{18}$$

$$b = |L'(\ell_b)| = \max_\ell |L'(\ell)|. \tag{19}$$

Finally, it calculates the sums

$$w_a = \sum_{\ell \in R'(j_a)} (v_{j_a}, v_\ell), \tag{20}$$

$$w_b = \sum_{j \in L'(\ell_b)} (v_j, v_{\ell_b}), \tag{21}$$

$$w_c = \sum_{j,\ell;\ L(j)=R(\ell)=\emptyset} (v_j, v_{\ell_b}). \tag{22}$$

Theorem 1 *For each transitive directed graph $D = (V, A)$, Algorithm 1 finds the class set C of a multiple classifier generated by some elements $g_1, \ldots, g_k \in F_D^m$ and such that C can correct the largest number of classification errors of individual binary classifiers. The running time of the algorithm is $O(n^2)$.*

A complete proof of our main theorem is rather nontrivial. It is included in a separate section below.

Thus, our paper has found the classifiers optimal from the point of view of their error correcting capability. As noted above there are several other additional properties necessary for creating efficient multiple classifiers.

4 Examples

This section contains examples of certain special classes of multiple classifiers $C(g_1, \ldots, g_k)$ considered in the main algorithm. The example demonstrates that the number of pairwise distinct class sets $C(g_1, \ldots, g_k)$ can grow exponentially with n. Therefore, a complete enumeration algorithm answering the same question as our quadratic algorithm would have exponential running time.

Example 2 shows that there are many class sets which can be generated only by several elements.

Example 2 Let $D = (V, A)$ be a directed graph with the set

$$V = \{v_1, \ldots, v_{2n}\}$$

of vertices and the set

$$A = \{(v_i, v_j) \mid 1 \leq i < j \leq 2n\}$$

of arcs. Evidently, the cardinality of the set A is equal to $m = |A| = n(2n-1)$. Let P be the set of all subsets of the set

$$V_{2n-1} = \{v_1, \ldots, v_{2n-1}\},$$

and let P_n be the set of n-element sets in P. Evidently, $|P| = 2^{2n-1}$ and $|P_n| = \binom{2n-1}{n} = \frac{(2n-1)!}{n!(n-1)!}$. For each subset $S \in P$, let us consider the set

$$g_S = \{(v_i, v_{i+1}) \mid v_i \in S\},$$

of elements in F^m. Looking at all elements of the set $g_S \subseteq F^m$ we can write them all down so that

$$g_S = \{g_1, \ldots, g_k\},$$

where $k = k(S) = |S|$. Let us consider the following class sets:

$$C(g_S) = C(g_1, \ldots, g_k)$$

Proposition 3 *In the notation of Example 2 the following properties hold:*

(i) *If $|S| > 1$, then the class set $C(g_S)$ cannot be generated by one element in F^m.*

(ii) *For all $S, T \in P$,*

$$C(g_S) \subseteq C(g_T) \Longleftrightarrow S \subseteq T.$$

(iii) *All class sets $C(g_S)$, for $S \in P_n$, are pairwise incomparable with respect to inclusion.*

Properties (i) and (iii) above mean that the number of pairwise incomparable class sets of the form $C(g_1, \ldots, g_k)$ grows exponentially. It follows that a brute force algorithm answering the same question as our quadratic algorithm has exponential running time.

5 Proofs

Proof of Theorem 1. First, let us find the running time of Algorithm 1. Clearly, lines 2, 3 take $O(n^2)$ time to execute. The running time of lines 4 to 9 is $O(n^2)$. The same can be said of lines 10 to 17. The cardinalities of the sets $L(j)$ and $R(j)$ can be calculated when these sets are collected, which has not been indicated in the algorithm explicitly just for brevity.

It follows that lines 18 and 19 can be done in $O(n)$ time. Lines 20 to 21 execute in $O(n^2)$ time. Each of the lines 28 and 32 requires $O(n)$ time. Therefore, the running time of Algorithm 1 is $O(n^2)$ in total.

Second, we are going to verify that the class set $C(w_a)$ can correct $\lfloor (a-1)/2 \rfloor$ classification errors. As noted above, the number of errors a multiple classifier can correct is determined by its Hamming weight: a class set can correct $\lfloor (a-1)/2 \rfloor$ classification errors if and only if it has Hamming weight at least a. Thus, it suffices to demonstrate that $C(w_a)$ has minimum distance a.

The definition (13) tells us that $C(w_a)$ contains w_a. By (18) and (20), we get $\mathrm{wt_H}(w_a) = a$. Hence it follows from the definition of the Hamming weight of a class set that $\mathrm{wt_H}(C(w_a)) \leq a$.

Notice that the way Algorithm 1 constructs w_a according to the rule (20) allows us to simplify the representation of $C(w_a)$ provided by (13). Namely, the condition $R(\ell) = 0$ verified before collecting ℓ in the set $R'(j_a)$ in (17) ensures that $R(i) = \emptyset$ for all $i \in R'(j_a)$. Hence it follows from (15) and (10) that

$$w_a F^m = \{w_a f \mid f \in F^m\} = 0. \tag{23}$$

Now we can simplify (13) by assuming that all $h_{j,1}$ are equal to 1, and then representing all the multipliers $f_{j,1}$ as one whole set F^m. Therefore we get

$$C(w_a) = F^m w_a + F w_a. \tag{24}$$

Choose any element y in the set $C(w_a)$ with minimum nonzero Hamming weight. Equalities (24) and (10) yield us

$$\begin{aligned} y &= r w_a + f w_a \\ &= \sum_{e \in A} f_e e w_a + f w_a \\ &= \sum_{s=1}^{t} f_s (v_{i_s}, v_{j_a}) w_a + f w_a, \end{aligned} \tag{25}$$

where $r \in F^m$, $f, f_e, f_s \in F$ and t is a positive integer. We may assume that this expression has been simplified by removing all zero summands and all terms which cancel out.

If all the indices $i_1, \ldots, i_t$ coincide with j_a in (25), then (25) implies that

$$y = r w_a,$$

for some $r \in F$. Hence we get $\mathrm{wt_H}(y) = a$ in this case.

On the other hand, if not all of these indices are equal to j_a, then there exists a nonzero summand of y involving the arc (v_{i_s}, v_{j_a}) in (25) such that $i_s \neq j_a$. Put $j = i_s$. It is easily seen from (10) that for the fixed given j these terms do not cancel with other summands where $i_s \neq j$. Besides, the sum of these terms taken separately belongs to $C(g_a)$ too by (13). Hence we could have chosen it from the very beginning as an element with a minimum Hamming weight in $C(g_a)$. Therefore by the minimality of y we get $y = f'(v_j, v_{j_a})w_a$, for some $f' \in F$. It follows that $\mathrm{wt_H}((v_j, v_{j_a})w_a) = \mathrm{wt_H}(w_a) = a$, again.

Thus, in each case we get $\mathrm{wt_H}(y) = a$. Therefore $\mathrm{wt_H}(C(w_a)) = a$, as claimed.

Third, let us consider the class set $C(w_b)$. By (11), we get $w_b \in C(w_b)$. Hence $\mathrm{wt_H}(C(w_b)) \leq b$, because $\mathrm{wt_H}(w_b) = b$.

Let us now prove the reversed inequality. It follows from (14), (16), (21) and (7) that

$$F^m w_b \;=\; \{f w_b \mid f \in F^m\} = 0. \tag{26}$$

Hence we can simplify (13) for $C(w_b)$ and get

$$C(w_b) \;=\; w_b F^m + F w_b. \tag{27}$$

Consider any nonzero element z in the set $C(w_b)$ with minimum Hamming weight. Equality (27) implies that

$$\begin{aligned} z &= w_b r + f w_b \\ &= \sum_{e \in A} f_e w_b e + f w_b \\ &= \sum_{s=1}^{t} f_s w_b(v_{\ell_b}, v_{i_s},) + f w_b, \end{aligned} \tag{28}$$

where $r \in F^m$, $f, f_e, f_s \in F$ and t is a positive integer. We may assume that this expression has already been simplified by removing all zero summands and all terms which cancel.

If all the indices $i_1, \ldots, i_t$ in (28) coincide with ℓ_b, then (10) implies that

$$z = w_b r,$$

for some $r \in F$. Hence we get $\mathrm{wt_H}(z) = b$ in this case.

If, however, not all of these indices coincide with ℓ_b, then there exists a nonzero summand of z involving the arc (v_{ℓ_b}, v_{i_s}) in (28) such that $i_s \neq \ell_b$.

Put $\ell = \ell_b$. It is easily seen from (10) that for the fixed given ℓ these terms do not cancel with other summands where $i_s \neq \ell$. Besides, the sum of these terms taken separately belongs to $C(g_b)$ too by (13). Hence the minimality of z implies that $z = f' w_b(v_{\ell_b}, v_\ell)$, for some $f' \in F$. It follows that $\mathrm{wt_H}(z) = \mathrm{wt_H}(w_b) = b$.

Thus, in both the cases above $\mathrm{wt_H}(z) = b$. This means that $\mathrm{wt_H}(C(w_b)) = b$. Therefore $C(w_b)$ can correct $\lfloor (b-1)/2 \rfloor$ classification errors.

Fourth, let us consider $C(w_c)$. It follows from (10), (11) and (22) that

$$F^m w_c = w_c F^m = 0. \tag{29}$$

Hence

$$C(w_c) = \sum_{s=1}^{c} F w_c. \tag{30}$$

Therefore the Hamming weight of $C(w_c)$ is equal to c, and so it can correct $\lfloor (c-1)/2 \rfloor$ classification errors.

Fifth, let us take any class set $C = C(g_1, \ldots, g_k)$. Denote the Hamming weight of C by d. We are going to demonstrate that one of the class sets $C(w_a)$, $C(w_b)$, $C(w_c)$ can always correct at least as many classification errors as C. In view of what we have already shown above, this is equivalent to saying that d does not exceed $\max\{a, b, c\}$.

Choose a nonzero element x with minimum Hamming weight d in C. By (4), we can represent it as

$$x = \sum_{s=1}^{d} f_s(v_{j_s}, v_{\ell_s}), \tag{31}$$

where $0 \neq f_s \in F$ for all s. We may assume that (31) has been simplified by combining the likely terms. Next, let us consider all cases that may occur.

Case 1. $|L(v_{j_t})| \neq 0$ and $|R(v_{\ell_t})| \neq 0$ for some t. Then we can pick i, i' such that $1 \leq i, i' \leq n$, the element v_i belongs to $L(v_{j_t})$ and the element $v_{i'}$ belongs to $R(v_{\ell_t})$. It follows from (14) and (15) that $(v_i, v_{j_t}), (v_{\ell_t}, v_{i'}) \in A$. By (10), we get

$$(v_i, v_{j_t}) x (v_{\ell_t}, v_{i'}) = f_t(v_i, v_{i'}). \tag{32}$$

Besides, (11) shows that the product (32) belongs to C too. Hence $\mathrm{wt_H}(C) = 1$ in this case, and so C cannot correct any classification errors. Thus the assertion is trivial in this case.

Case 2. $|L(v_{j_t})| \neq 0$ for some $1 \leq t \leq d$, but $|R(v_{\ell_t})| = 0$. Then we can find j such that $1 \leq j \leq n$ and the element v_j belongs to $L(v_{j_t})$. By (11),

the product $(v_j, v_{j_t})x$ belongs to C too. Since this product is nonzero, the minimality of d implies that all terms $(v_j, v_{j_t})(v_{j_s}, v_{\ell_s})$ of $(v_j, v_{j_t})x$ have to be nonzero. Hence (10) shows that

$$i_1 = \cdots = i_d = i_t.$$

If $|R(v_{\ell_s})| \neq 0$ for some s, then we are in the conditions of Case 1 again. Hence further we may assume that $|R(v_{\ell_s})| = 0$ for all s. This means that $v_{\ell_s} \in R'(i_t)$ for all s. By the maximality of a, we get $a \geq |R'(i_t)|$. It follows that $C(w_a)$ can correct at least as many classification errors as C in this situation.

Case 3. $|L(v_{j_s})| = 0$ for all $s = 1, \ldots, d$, and $|R(v_{\ell_t})| \neq 0$ for some t, where $1 \leq t \leq d$.

Then there exists a vertex v_j in $R(v_{\ell_t})$. By (11), the product $x(v_{\ell_t}, v_j)$ belongs to C as well. By the minimality of d, it is clear that all summands $(v_{j_s}, v_{\ell_s})(v_{\ell_t}, v_j)$ of $x(v_{\ell_t}, v_j)$ are all nonzero, because otherwise the Hamming weight of this product would be less than d. Therefore (10) implies that $v_{\ell_s} = v_{j_t}$ for all s. Hence $j_s \in L(v_t)$ for all $s = 1, \ldots, d$. In this case (16) guarantees that $v_{j_s} \in L'(v_{\ell_t})$ for all s. Hence the maximality of b shows that $b \geq |L'(v_{\ell_t})| = d$. Therefore $C(w_b)$ can correct at least as many classification errors as C in this case.

Case 4. $|L(v_{j_s})| = |R(v_{j_t} y_{v_t})| = 0$ for all $s = 1, \ldots, d$. Then all terms involving (v_{j_s}, v_{ℓ_s}) are summands of w_c. Therefore in this case $C(w_c)$ can correct at least as many classification errors as C does.

This completes the proof of correctness of Algorithm 1. □

Proof of Proposition 3. Condition (iii) immediately follows from (ii). In proving (i) and (ii) we are going to use the set

$$T = \{(c, d) \mid c + 1 < d\}$$

and the set I_T of all finite sums of the elements of T.

(i): Suppose to the contrary that $|S| > 1$, but there exists an element $r \in F^m$ generating the whole class set $C(g_S)$ so that $C(r) = C(g_S)$.

Since $|S| \geq 2$, evidently, there exist a and b such that $a \neq b$, $1 \leq a, b \leq n - 1$ and $v_a, v_b \in S$. By the definition of g_S, we get

$$(a, a + 1), (b, b + 1) \in g_S. \tag{33}$$

We can rewrite condition (11) so that it describes the elements of the class set generated by r. Using this and noting that $g_S \subseteq C(g_S) = C(r)$,

implies $(a, a+1) \in C(r)$, we see that there exist

$$f_j, h_j \in \{f(u,v) \mid f \in F, (u,v) \in A\} \cup \{1\} \tag{34}$$

such that

$$(a, a+1) \;=\; \sum_{j=1}^{m_1} f_j r h_j. \tag{35}$$

Let us now compare the products of (35) with the set T. Take any nonzero product $f_j r h_j$ in (35). If $f_j \neq 1$, then it easily follows from (7) that $f_j r h_j \in T$. Similarly, if $h_j \neq 1$, then (7) implies that $f_j r h_j \in T$. Since $(a, a+1) \notin T$, we see that the only way to obtain $(a, a+1)$ in (35) is to take $f_j = h_j = 1$. Then it is clear that $(a, a+1)$ is a summand of r. Likewise, we can also prove that $(b, b+1)$ is a summand of r too. However, this shows that for the term with $f_j = h_j = 1$ both $(a, a+1)$ and $(b, b+1)$ simultaneously occur in the same summands of (35). Since $a \neq b$, we get a contradiction with (35).

(ii): If $S \subseteq T$, then by definition it is obvious that $C(g_S) \subseteq C(g_T)$. Conversely, let us assume that $C(g_S) \subseteq C(g_T)$ and suppose to the contrary, that $S \not\subseteq T$.

Choose any $s \in S \setminus T$. The same argument as in the proof of (i) demonstrates that the inclusion $(s, s+1) \in C(g_T)$ implies that there exist

$$\begin{aligned} t_i \quad &\in T, \\ f_{i,j}, h_{i,j} \quad &\in \{f(u,v) \mid f \in F, (u,v) \in A\} \cup \{1\} \end{aligned} \tag{36}$$

such that

$$(s, s+1) \;=\; \sum_{i=1}^{\ell} \sum_{j=1}^{m_1} f_{i,j}(t_i, t_i+1) h_{i,j}. \tag{37}$$

Again we compare the products of (37) with the set T. As before, it follows from (7) that $f_{i,j} r_i h_{i,j} \in T$ if $f_{i,j} \neq 1$ or $h_{i,j} \neq 1$. Given that $(s, s+1) \notin T$, the only way of obtaining $(s, s+1)$ in (37) is to take $f_j = h_j = 1$. Then it is clear that $(s, s+1) = (t_i, t_i+1)$ for some $1 \leq i \leq \ell$. Hence $s = t_i \in T$. This contradiction completes the proof. □

6 Example of an application

Following [13], here we include an example of a practical application just to illustrate. We consider the problem of classifying all letters in the letter

dataset from [4]. Here are possible approaches to solving this classification problem.

1. Separate binary classifier for each letter. Using letters of the letter dataset from [4], for each letter x of the alphabet A, one can train a separate neural network $N[x]$ to recognize x. It will recognize the letter x with approximately 80% accuracy. Then there are two options for combining these binary classifiers.

1.1. Majority multiple classifier. Given an arbitrary input y from the letter dataset, the Majority Multiple Classifier classifies y according to which of the binary classifiers $N[x]$, $x \in A$, produces the largest output and so is most confident in recognizing y as the letter it has been trained to recognize.

1.2. A neural network multiple classifier. A new neural network can be trained to combine binary classifiers $N[x]$, $x \in A$, to classify arbitrary input y from the letter dataset. This may improve the efficiency, since it is likely that the outputs of the neural networks of similar letters correlate.

2. Binary classifiers for groups of letters.

2.1. Digits in the alphabet. To illustrate the method, let us number all letters according to their order in the alphabet and consider these numbers in binary notation to form groups of letters. Let us denote by G_i the group of all letters with i-th last digit in binary notation equal to 1. Thus A has number 0 and does not belong to any group, B has number 1 and belongs to the groups G_1 only. One could train neural networks N_1, N_2, N_3, N_4, N_5 to recognize the groups G_1, G_2, G_3, G_4, G_5. Given the outputs of N_1, N_2, N_3, N_4, N_5 for an arbitrary letter, the number of the letter in the alphabet then is given by $N_1 + 2 * N_2 + 4 * N_3 + 8 * N_4 + 16 * N_5$. We see that only 5 binary classifiers may be sufficient to construct a multiple classifier for all letters.

2.2. Groups based on similarity. It is likely that the neural networks trained to recognize groups based on the order in the alphabet will be less accurate, because these groupings are unrelated to the similarity in the appearances of the letters. It would be nice to find five groups $G1$, $G2$, $G3$, $G4$, $G5$ of letters such that all letters in each group are similar and

each letter can be determined uniquely according to its membership in the groups. This would improve the accuracy of the neural networks trained as binary classifiers to recognize the membership of letters in these groups.

2.3. Genetic algorithm for improving groups. In order to look at all groups like this, it is enough to order all letters in a different order and then number them again and use the same division according to the binary digits in the new numbering. If we want to include the option of groups with unequal sizes, then we can add one more classifier $N6$, and allow gaps or spaces in the ordering of letters before numbering them. In order to find which of the groupings achieves better efficiency, one could use genetic algorithms and evolve the population of several groupings or orderings.

Of course, this process is very computationally demanding. It is however likely to produce some efficient groupings allowing the neural networks to be trained to reasonably high levels of accuracy.

3. Error correction. Now that there are only 5 binary classifiers being combined, we can use an error correcting code and add a few neural networks recognizing additional check digits. This could significantly improve the accuracy of the multiple classifier.

Experimental investigation of these methods is very computationally expensive and can be addressed in separate publications.

7 Open questions

Our paper makes the very first step in the study of the applications of combinatorial algorithms based on directed graphs to constructions of multiple classifiers for data mining. There are many interesting questions which remain open.

Let us begin with a general problem. The method proposed in the present paper relies on additional operations defined on the class set. This is in fact motivated by the way cyclic codes are defined in coding theory. On the other hand, it is well known that there are also efficient codes, which are not cyclic. This motivated the following problem.

Problem 4 *Introduce methods of defining multiple classifiers which do not depend on properties of additional operations on the class set.*

The techniques of this paper make it possible to believe that the following question can be answered in the nearest future.

Problem 5 *For each positive integer n, find conditions on graphs $D = (V, A)$ with $|V| = n$ necessary and sufficient for there to exist a classifier $C(g_1, \ldots, g_m)$ in F_D^n which is able to correct the largest number of classification errors among all classifiers of this type for the given n.*

Problem 6 *Suppose that for some m, k there exists a multiple classifier correcting k errors of m binary classifiers. Is it always possible to find a graph D such that there is a classifier $C(g_1, \ldots, g_m)$ in F_D^n correcting at least k errors of m binary classifiers?*

It would be interesting to determine whether in our search for optimal classifiers it is enough to consider some important and special classes of labeled graphs, or graphs satisfying additional properties. In particular, we refer to [34] for the previous results on Moore graphs and Moore bound.

Problem 7 *Can we find directed graphs as close as possible to optimality in terms of the Moore bound and at the same time defining efficient multiple classifiers?*

8 Acknowledgements

The authors are grateful to the organisers of the International Workshop on Combinatorial Algorithms IWOCA2007. The first author was supported by Australian Research Council, Discovery grant DP0449469, and a RIBG grant of the University of Ballarat. The third author was supported by Queen Elizabeth II Fellowship and Discovery grant DP0211866 from Australian Research Council.

References

[1] R. Alfaro, A.V. Kelarev, Recent results on ring constructions for error-correcting codes, *Contemporary Math.* 376(2005), 1–12.

[2] R. Alfaro, A.V. Kelarev, On cyclic codes in incidence rings, *Studia Sci. Math. Hungarica* 43(2006)(1), 69–77.

[3] I.M. Araújo, A.V. Kelarev, A. Solomon, An algorithm for commutative semigroup algebras which are principal ideal rings with identity, *Comm. Algebra* 32(2004)(4), 1237–1254.

[4] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, Irvine, CA: University of California, [http://www.ics.uci.edu/ mlearn/MLRepository.html], [http://mlearn.ics.uci.edu/MLRepository.html], Dept. Information & Computer Sci., 2007.

[5] M. Bača, S. Jendroľ, M. Miller, J. Ryan, Antimagic labelings of generalized Petersen graphs that are plane, *Ars Combin.* 73(2004), 115–128.

[6] A.M. Bagirov, J.L. Yearwood, A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems, *European J. Operational Research* 170(2006), 578–596.

[7] A.M. Bagirov, A.M. Rubinov, J. Yearwood, A global optimization approach to classification, *Optim. Eng.* 3(2002), 129–155.

[8] A.M. Bagirov, A.M. Rubinov, N.V. Soukhoroukova, J. Yearwood, Unsupervised and supervised data classification via nonsmooth and global optimization, *Top* 11(2003), 1–93.

[9] L. Brankovic, A. Rosa, J. Širán, Labellings of trees with maximum degree three—an improved bound, *J. Combin. Math. Combin. Comput.* 55(2005), 159–169.

[10] J. Cazaran, A.V. Kelarev, Generators and weights of polynomial codes, *Arch. Math. (Basel)* 69(1997), 479–486.

[11] J. Cazaran, A.V. Kelarev, S.J. Quinn, D. Vertigan, An algorithm for computing the minimum distances of extensions of BCH codes embedded in semigroup rings, *Semigroup Forum* 73(2006), 317–329.

[12] R. Downey, M.R. Fellows, G. Whittle, A. Vardy, The parameterized complexity of some fundamental problems in coding theory, *SIAM J. Comput.* 29(2)(1999), 545–570.

[13] B. Ferguson, R. Ghosh, J.L. Yearwood, Modular neural network design for the problem of alphabetic character recognition, *Internat. J. Pattern Recognition & Artificial Intelligence* 19(2006)(2), 249–269.

[14] M.T. Goodrich, R. Tamassia, *Data Structures and Algorithms in Java* (Monographs in Computer Science, Wiley, 2001).

[15] J.A. Gallian, Graph labeling, *Electronic J. Combinatorics*, Dynamic Survey DS6, January 20, 2005, 148pp.

[16] J. Holub, C.S. Iliopoulos, B. Melichar, L. Mouchard, Distributed string matching using finite automata, Australasian Workshop on Combinatorial Algorithms, AWOCA 99, Perth, 114–127.

[17] C.S. Iliopoulos, Computing in general abelian groups is hard, *Theoret. Comput. Sci.* 41(1985)(1), 81–93.

[18] B.H. Kang, A.V. Kelarev, A.H.J. Sale, R.N. Williams, A new model for classifying DNA code inspired by neural networks and FSA, *Lect. Notes Computer Science* 4303(2006), 187–198.

[19] A.V. Kelarev, *Ring Constructions and Applications* (World Scientific, River Edge, 2002).

[20] A.V. Kelarev, *Graph Algebras and Automata* (Marcel Dekker, New York, 2003).

[21] A.V. Kelarev, Labelled Cayley graphs and minimal automata, *Australasian J. Combinatorics* 30(2004), 95–101.

[22] A.V. Kelarev, An algorithm for BCH codes extended with finite state automata, *Fundamenta Informaticae* 84(2008)(2), 51–60.

[23] A.V. Kelarev, B.H. Kang, Dot Steane, Clustering algorithms for ITS sequence data with alignment metrics, *Lect. Notes Artificial Intelligence* 4304(2006), 1027–1031.

[24] A.V. Kelarev, M. Miller, O.V. Sokratova, Languages recognized by two-sided automata of graphs, *Proc. Estonian Akademy of Science* 54(2005)(1), 46–54.

[25] Kelarev, A.V. and D.S. Passman, A description of incidence rings of group automata, *Contemporary Mathematics* 456(2008), 27–33.

[26] A.V. Kelarev, C.E. Praeger, On transitive Cayley graphs of groups and semigroups, *European J. Combinatorics* 24(2003)(1), 59–72.

[27] A.V. Kelarev, S.J. Quinn, A combinatorial property and power graphs of groups, *Contrib. General Algebra* 12(2000), 229–235.

[28] A.V. Kelarev, J. Ryan, J.L. Yearwood, Cayley graphs as classifiers for data mining: the influence of asymmetries, *Discrete Math.*, accepted.

[29] A.V. Kelarev, O.V. Sokratova, Directed graphs and syntactic algebras of tree languages, *J. Automata, Languages & Combinatorics* 6(2001)(3), 305–311.

[30] A.V. Kelarev, O.V. Sokratova, On congruences of automata defined by directed graphs, *Theoretical Computer Science* 301(2003), 31–43.

[31] G. Luger, *Artificial Intelligence. Structures and Strategies for Complex Problem Solving*, 5th edition (Addison-Wesley, London, 2005).

[32] B.D. McKay, Transitive graphs with fewer than twenty vertices, *Mathematics of Computation*, 33(1979), 1101–1122.

[33] M. Miller, D. Patel, J. Ryan, K.A. Sugeng, S. Slamin, M. Tuga, Exclusive sum labeling of graphs, *J. Combin. Math. Combin. Comput.* 55(2005), 137–148.

[34] M. Miller, J. Širán, Moore graphs and beyond: A survey of the degree/diameter problem, *Electronic J. Combinatorics*, Dynamic Survey, DS14, December 5, 2005, 61pp.

[35] B. Smyth, *Computing Patterns in Strings* (Addison-Wesley, London, 2003).

[36] M. Tuga, M. Miller, J. Ryan, Z. Ryjáček, Exclusive sum labelings of trees, *J. Combin. Math. Combin. Comput.* 55(2005), 109–121.

[37] I.H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition (Elsevier/Morgan Kaufman, Amsterdam, 2005).

[38] J.L. Yearwood, M. Mammadov, *Classification Technologies: Optimization Approaches to Short Text Categorization* (Idea Group Inc., 2007).