Solving for multi-class using orthogonal coding matrices

Peter Mills[1]

Abstract

A common method of generalizing binary to multi-class classification is the error correcting code (ECC). ECCs may be optimized in a number of ways, for instance by making them orthogonal. Here we test two types of orthogonal ECCs on seven different datasets using three types of binary classifier and compare them with three other multi-class methods: 1 versus 1, one-versus-the-rest and random ECCs. The first type of orthogonal ECC, in which the codes contain no zeros, admits a fast and simple method of solving for the probabilities. Orthogonal ECCs are always more accurate than random ECCs as predicted by recent literature. Improvments in uncertainty coefficient (U.C.) range between 0.4 and 17.5% (0.004–0.139, absolute), while improvements in Brier score between 0.7 and 10.7%. Unfortunately, orthogonal ECCs are rarely more accurate than 1 versus 1. Disparities are worst when the methods are paired with logistic regression, with orthogonal ECCs never beating 1 versus 1. When the methods are paired with SVM, the losses are less significant, peaking at 1.5%, relative, 0.011 absolute in uncertainty coefficient and 6.5% in Brier scores. Orthogonal ECCs are always the fastest of the five multi-class methods when paired with linear classifiers. When paired with a piecewise linear classifier, whose classification speed does not depend on the number of training samples, classifications using orthogonal ECCs were always more accurate than the other methods and also faster than 1 versus 1. Losses against 1 versus 1 here were higher, peaking at 1.9% (0.017, absolute), in U.C. and 39% in Brier score. Gains in speed ranged between 1.1% and over 100%. Whether the speed increase is worth the penalty in accuracy will depend on the application.

1 Introduction

Many methods of statistical classication can only discriminate between two classes. Examples include lineear classifiers such as perceptrons and logistic regression [1], piecewise linear classifiers [2, 3], as well as support vector machines [4]. There are many ways of generalizing binary classification to multi-class and the number of possibilities increases exponentially with the number of classes.

One should distinguish between multi-class methods that use only a subset of the binary classifiers, adding more as the algorithm narrows down the class, and those that use all of the binary classifiers, combining the results or solving for the class probabilities. In the former category, we have hierarchical multi-class classifiers such as decision trees [5, 6] and decision directed acyclic graphs (DDACs) [7]. In the latter category, two common methods are one-versus-one (1 vs. 1) and one-versus-the-rest (1 vs. rest) [8]. These in turn generalize to error-correcting codes (ECCs) [9].

Early experiments with ECCs used random codes: the assumption is that if the codes are long enough (there are enough binary classifiers) they will adequately span the classes. Later work focused on optimizing the design of the codes: what type of codes will best span the classes and produce the most accurate results? Here

✉ Peter Mills | [1]Cumberland, Canada.

we can also distinguish between two types: those that use the data to help design the codes [10–12] and those that are independent of the data but use the mathematical properties of the codes themselves to aid in their construction [13–15]. It is these latter type of optimized error-correcting codes we turn to in this note.

In error-correcting coding, there is a coding matrix, A, that specifies how the set of multiple classes is partitioned for each binary classifier. For a given column, if members of the jth class are to be labeled $-1/+1$ for the binary classifier, then the jth row is assigned a $-1/+1$. If the jth class is left out, then the jth row is assigned a 0. Typically, the class of the test point is determined by the distance between a row in the matrix and a vector of binary *decision functions*:

$$c(\mathbf{x}) = \arg\min_i |\mathbf{a}_i - \mathbf{r}(\mathbf{x})| \qquad (1)$$

where $\mathbf{a}_i \in \{-1, 0, +1\}$ is the ith row of the coding matrix and $\mathbf{r}$ is a vector of decision functions at *test point*, $\mathbf{x}$. If we take the upright brackets as a Euclidean distance we can expand (1) as follows:

$$c = \arg\min_i \sum_j \left(|\mathbf{a}_i| + |\mathbf{r}| - 2\mathbf{a}_i \cdot \mathbf{r} \right)$$

Since $|\mathbf{r}|$ is constant over i, it may be removed from the expression. Also, for the purposes of this note, each row of A will be given the same number of non-zero entries, hence:

$$|\mathbf{a}_i| = |\mathbf{a}_j| = const.$$

This is most evident for the case in which each binary classifier partitions all of the classes so that there are no zeros in A as is the case for the one-versus-the-rest partitioning. Then (1) reduces to a *voting* solution:

$$c = \arg\max A\mathbf{r} \qquad (2)$$

Both [13] and [14] show that to maximize the accuracy of an ECC, the distance between each row, $|\mathbf{a}_i - \mathbf{a}_j|_{i\neq j}$, should be maximized. Using the above assumptions, this reduces to:

$$\min |\mathbf{a}_i \cdot \mathbf{a}_j|_{i\neq j}$$

Note the absolute value prevents degenerate rows. In other words, the coding matrix, A, should be orthogonal.

In this note, we describe a fast and simple algorithm that uses orthogonal ECCs to solve for the conditional probabilites in multi-class classification. There are three reasons to require the conditional probabilities:

1. Probabilities provide useful extra information, specifically how accurate a given classification is, in absence of knowledge of its true value.

2. The relationship between the binary probabilities and the multi-class probabilities derives uniquely and rigorously from probability theory.
3. Binary classifiers that do not return calibrated probability estimates, but nonetheless supply a continuous decision function, are easy to recalibrate so that the decision function more closely resembles a probability [16, 17].

Two types of orthogonal ECCs along with three other multi-class methods-1 versus 1, 1 versus the rest, and random ECCs–will be tested on seven different datasets using three different binary classifiers–logistic regression, support vector machines (SVM), and piece-wise linear–to see how they compare in terms of classification speed, classification accuracy and accuracy of the conditional probabilities.

2 Algorithm

We wish to design a set of m binary classifiers, each of which return a decision function:

$$r_j(\mathbf{x}) = P_j(-1|\mathbf{x}) - P_j(+1|\mathbf{x})$$

where $P_j(c|\mathbf{x})$ is the conditional probability of the cth class of the jth classifier. Each binary classifier partitions a set of m classes such that for a given test point, $\mathbf{x}$:

$$\sum_{i=1}^{m} a_{ij} p_i = r_j; \quad j = [1, \ldots, n]$$

where $A = \{a_{ij} \in \{-1, +1\}\}$ is a *coding matrix* for which each code partitions all of the classes and $p_i = p(i|\mathbf{x})$ is the conditional probability of the ith class. In vector notation:

$$A^T \mathbf{p} = \mathbf{r} \qquad (3)$$

This result derives from the fact that the class probabilities are additive [18]. The more general case where a class can be excluded, that is the coding may include zeroes, $a_{ij} \in \{-1, 0, +1\}$, will be treated in the next section.

Note that this assumes that the binary decision functions, $\mathbf{r}$, estimate the conditional probabilities perfectly. In practice there are a set of constraints that must be enforced because $\mathbf{p}$ is only allowed to take on certain values. Thus, we wish to solve the following minimization problem:

$$\arg\min_{\mathbf{p}} |A^T \mathbf{p} - \mathbf{r}| \qquad (4)$$

$$\sum_{i=1}^{m} p_i = 1 \qquad (5)$$

$$p_i \geq 0; \quad i = [1, \ldots, m] \qquad (6)$$

If A is orthogonal,

$$AA^T = nI$$

where I is the $m \times m$ identity matrix, then the unconstrained minimization problem is easy to solve. Note that the voting solution in (2) is now equivalent to the inverse solution in (3). This allows us to determine the class easily, but we also wish to solve for the probabilities, $\mathbf{p}$, so that none of the constraints in (5) or (6) are violated.

The orthogonality property allows us to reduce the minimization problem in (4) to something much simpler:

$$\arg \min_{\mathbf{p}} |\mathbf{p} - \mathbf{p}_0|$$

where $\mathbf{p}_0 = A\mathbf{r}/n$ with the constraints in (5) and (6) remaining the same. Because the system has been rotated and expanded, the non-negativity constraints in (6) remain orthogonal, meaning they are independent: enforcing one by setting one of the probabilities to zero, $p_k = 0$ for example, shouldn't otherwise affect the solution. This still leaves the normalization constraint in (5): the problem, now strictly geometrical, is comprised of finding the point nearest p_0 on the diagonal hyper-surface that bisects the unit hyper-cube.

Briefly, we can summarize the algorithm as follows: (1) move to the nearest point that satisfies the normalization constraint, (5); (2) if one or more of the probabilities is negative, move to the nearest point that satisfies both the normalization constraint and the non-negativity constraints, (6), for the negative probabilities; (3) repeat step 2. More formally, let $\mathbf{1}$ be a vector of all 1's:

- $i := 0; m_0 := m$
- while $\exists k\, p_{ik} < 0 \vee \mathbf{p}_i \cdot \mathbf{1} \neq 1$:

    - if $\mathbf{p}_i \cdot \mathbf{1} \neq 1$ then $\mathbf{p}_{i+1} := \mathbf{p}_i + (\mathbf{p}_i \cdot \mathbf{1} - 1)/m_i$
    - let K be the set of k such that $p_{i+1,k} < 0$
    - for each $k \in K$:

        - $p_k := 0$
        - Remove k from the problem

    - $m_{i+1} := m_i - |K|$
    - $i := i + 1$

Note that resultant direction vectors for each step form an orthogonal set. For instance, suppose $m_0 = 4$ and after enforcing the normalization constraint, the first probability is less than zero, $p_{1,1} < 0$, then the direction vectors for the two motions are:

$$\frac{1}{2}[1,1,1,1] \cdot \frac{1}{2\sqrt{3}}[-3,1,1,1] = 0$$

More generally, consider the following sequence of vectors:

$$v_{ij} = \frac{1}{\sqrt{(m-i)^2 - 2(m-i-1)}} \begin{cases} 0; & j < i \\ -m+i+1; & j = i \\ 1; & j > i \end{cases}$$

where $i \in [1, m]$ and $j \in [1, m]$. [19] A nice feature of this method, in addition to being fast, is that it is divided into two stages: a solution stage and a normalization stage.

3 Constructing the coding matrix

Finding an A such that $AA^T = nI$ and $a_{ij} \in \{-1, 1, \}$ is quite a difficult combinatorial problem. When zeros are added in, $a_{ij} \in \{-1, 0, 1\}$, it becomes even more difficult. Work in signal processing may be of limited applicability because coding matrices are typically comprised of 0's and 1's rather than -1's and $+1$'s [20, 21]. In our case, a further restriction is that columns must contain both positive and negative elements, or:

$$\sum_{i=0}^{m} a_{ij} \neq \sum_{i=0}^{m} |a_{ij}|; \quad j = [1 \dots n] \tag{7}$$

A simple method of designing an orthogonal A is using harmonic series. Consider the following matrix for six classes ($m = 6$) and eight binary classifiers ($n = 8$):

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$
$$\tag{8}$$

This will limit the size of m relative to n; more precisely: $m \leq \lfloor 2\log_2 n \rfloor$. Moreover, only certain values of n will be admitted: $n = 2^t$ where t is a whole number.

The first three rows in (8) comprise a Walsh-Hadamard code [22]: all possible permutations are listed. A square ($n = m$) orthogonal coding matrix is called a Hadamard matrix [23]. It can be shown that besides $n = 1$ and $n = 2$, only Hadamard matrices of size $n = 4t$ exist, and it is still unproven that examples exist for all values of t [24]. A very simple, recursive method exists to generate matrices of size $n = t^2$ [24] but cannot be made to have the property in (7) since the matrix includes both a row and column of only ones. Such a matrix will include a "harmonic series" of the same type as in (8).

Two types of orthogonal coding matrices are tested in this note. The first type includes no zeros and is generated using a "greedy" algorithm. We choose n to be the smallest multiple of 4 equal to or larger than m. and start with an empty matrix. Candidate vectors containing both positive and negative elements are chosen at random to comprise a row of the matrix but never repeated. If the candidate vector is orthogonal to existing rows, then it is added to the matrix. New candidates are tested until the matrix is filled or we run out of permutations. A full matrix is almost always returned especially if $m < n$. The matrix is then checked to ensure that each column contains both positive and negative elements. Note that the whole process can be repeated as many times as necessary. An eight-class example follows:

$$A = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}$$

This type of coding matrix can be solved using the algorithm described in Sect. 2, above.

The other type of orthogonal coding matrix to be tested in this note includes zeros. The construction is similar except now the matrix is allowed to take on values of zero while the number of non-zero values (-1 or $+1$) is kept fixed. A size is chosen for the matrix typically larger than the number of classes while the resulting matrix will normally be somewhat smaller since degenerate and fixed value columns (a correctly-trained binary classifier would always return the same value) are removed. The parameters chosen for each class size are shown in Table 1.

Coding matrices of this type were generated by pure, brute force with no attempt to track previous trials. An example coding matrix for six classes is shown below. Redundant columns have been bold out.

$$A = \begin{bmatrix} -1 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{-1} \\ 0 & 1 & -1 & 0 & -1 & -1 & 0 & -1 & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & \mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{-1} \\ 1 & 0 & 0 & -1 & 0 & -1 & -1 & 1 & \mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \\ 0 & -1 & -1 & 0 & -1 & 1 & -1 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & -1 & 0 & 1 & 0 & -1 & 1 & 0 & \mathbf{0} & \mathbf{-1} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

This type of orthogonal ECC is solved using a general, iterative, constrained, linear least-squares solver [25].

Table 1 Table showing parameters chosen for the second type of orthogonal coding matrix: for the number of classes, m, the initial length of the code, n_0, and the number of non-zero values in each code, $|\mathbf{a}_i|$ ($i = 1, \ldots, m$), are given

m	n_0	$|\mathbf{a}_i|$
4	7	4
6	12	6
7	15	7
8	17	8
9	20	9
10	23	10

$n_0 \approx m \log_2 m$

More work will need to be done to find efficient methods of generating these matrices if they are to be applied efficiently to problems with a large number of classes.

4 Results

Orthogonal error-correcting codes were tested on seven different datasets: two for digit recognition–"pendigits" [26] and "usps" [27]; the space shuttle control dataset–"shuttle" [28]; an urban land classification dataset–"urban" [29]; a similar one for satellite land classification–"sat"; a dataset for patterned image recognition–"segment"; and a dataset for vehicle recognition–"vehicle" [30]. The last three are borrowed from the "statlog" project [1, 28].

Two types of orthogonal ECCs were tested: the first type described in Sect. 3, with no zeros in the codes, and the second type which includes zeros. These were compared with three other methods: one-versus-one, one-versus-the-rest, and random ECCs with the same length of coding vector (number of columns), m, as the orthogonal matrices of the first type. The 1 versus rest multi-class as well as the random ECCs were solved using the same type of constrained linear least squares method as used for the second type of orthogonal ECC [25]. By enforcing the normality constraints using a Lagrange multiplier, 1 versus 1 may be solved with a simple (unconstrained) linear equation solver [31].

Three types of binary classifier were used: logistic regression [1], support vector machines [4], and a peice-wise-linear classifer [3]. Logistic regression classifiers were trained using LIBLINEAR [32].

Support vector machines (SVMs) were trained using LIBSVM [33]. Partitions were trained separately then combined by finding the union of sets of support vectors for each partition. By indexing into the combined list of support vectors, the algorithms are optimized in both space and time [33]. For SVM, the same parameters were used for all multi-class methods and for all partitions (matrix columns). All datasets were trained using "radial basis function" (Gaussian) kernels of differing widths.

LIBSVM was also used to train an intermediate model from which an often faster piecewise-linear classifier [3] was trained. It was thought that this classifier would provide a better use-case for orthogonal ECCs than either of the other two. The single parameter for this algorithm–the number of border vectors–was set the same for each dataset as used in [3] for the 1 versus 1. For the other multi-class algorithms, the number of border vectors was doubled for small values (under 100) and increased by fifty percent for larger values to account for the more complex decision function created by using more classes in each binary classifier. Multi-class classifiers were designed,

Table 2 Total classification time, solution time, uncertainty coefficient and Brier score for seven different datasets using five different coding matrices: 1 versus 1, 1 versus the rest, randoms, orthogonal with no zeros, and orthogonal with zeros

Dataset	Method	Time (s)	Sol. only (s)	U.C.	Brier score
pendigits	1 versus 1	0.489 ± 0.006	0.410 ± 0.004	**0.956 ± 0.006**	**0.0566 ± 0.003**
	1 versus rest	0.118 ± 0.0042	0.0823 ± 0.0011	0.864 ± 0.008	0.113 ± 0.002
	ECC	0.18 ± 0.01	0.136 ± 0.007	0.723 ± 0.026	0.180 ± 0.008
	Ortho. 1	**0.048 ± 0.004**	**0.01095 ± 8e−5**	0.785 ± 0.010	0.172 ± 0.002
	Ortho. 2	0.24 ± 0.01	0.185 ± 0.010	0.862 ± 0.010	0.123 ± 0.009
sat	1 versus 1	0.092 ± 0.004	0.067 ± 0.001	**0.736 ± 0.009**	**0.176 ± 0.004**
	1 versus rest	0.033 ± 0.0048	0.0202 ± 2e−4	0.677 ± 0.007	0.204 ± 0.002
	ECC	0.043 ± 0.0048	0.0274 ± 6e−4	0.637 ± 0.025	0.217 ± 0.009
	Ortho. 1	**0.019 ± 0.006**	**0.00422 ± 8e−5**	0.665 ± 0.009	0.210 ± 0.002
	Ortho. 2	0.046 ± 0.005	0.0271 ± 0.0017	0.688 ± 0.018	0.197 ± 0.010
segment	1 versus 1	0.04 ± 5.9e−06	0.0336 ± 4e−4	**0.911 ± 0.009**	**0.0987 ± 0.0057**
	1 versus rest	0.012 ± 0.0042	0.0094 ± 2e−4	0.868 ± 0.010	0.144 ± 0.004
	ECC	0.016 ± 0.0052	0.0124 ± 4e−4	0.803 ± 0.040	0.179 ± 0.020
	Ortho. 1	**0.004 ± 0.005**	**0.00168 ± 6e−5**	0.849 ± 0.015	0.166 ± 0.004
	Ortho. 2	0.02 ± 2.9e−06	0.0147 ± 0.0012	0.880 ± 0.018	0.127 ± 0.008
shuttle	1 versus 1	1.10 ± 0.03	0.867 ± 0.014	**0.796 ± 0.013**	**0.0824 ± 0.0017**
	1 versus rest	0.33 ± 0.01	0.185 ± 0.003	0.605 ± 0.010	0.1341 ± 0.0006
	ECC	0.42 ± 0.01	0.265 ± 0.011	0.535 ± 0.120	0.144 ± 0.026
	Ortho. 1	**0.183 ± 0.005**	**0.042 ± 0.001**	0.593 ± 0.006	0.131 ± 0.002
	Ortho. 2	0.48 ± 0.03	0.31 ± 0.03	0.710 ± 0.095	0.101 ± 0.024
urban	1 versus 1	0.031 ± 0.003	0.0185 ± 1e−4	0.693 ± 0.026	**0.188 ± 0.006**
	1 versus rest	**0.007 ± 0.005**	0.0052 ± 4e−4	0.667 ± 0.018	0.204 ± 0.004
	ECC	0.009 ± 0.003	0.0068 ± 4e−4	0.647 ± 0.031	0.210 ± 0.008
	ortho. 1	**0.007 ± 0.005**	**0.00064 ± 4e−5**	0.674 ± 0.016	0.206 ± 0.004
	ortho. 2	0.014 ± 0.005	0.0082 ± 6e−4	**0.693 ± 0.017**	0.198 ± 0.006
usps	1 versus 1	0.63 ± 0.01	0.347 ± 0.005	**0.898 ± 0.010**	**0.0827 ± 0.0022**
	1 versus rest	0.152 ± 0.004	0.0704 ± 9e−4	0.840 ± 0.007	0.112 ± 0.003
	ECC	0.205 ± 0.005	0.112 ± 0.005	0.769 ± 0.021	0.1416 ± 0.006
	Ortho. 1	**0.1 ± 2.1e−05**	**0.0096 ± 5e−4**	0.815 ± 0.009	0.132 ± 0.002
	Ortho. 2	0.30 ± 0.02	0.16 ± 0.01	0.846 ± 0.015	0.112 ± 0.004
vehicle	1 versus 1	0.002 ± 0.004	0.00436 ± 8e−5	**0.685 ± 0.041**	**0.245 ± 0.011**
	1 versus rest	0	0.00142 ± 6e−5	0.654 ± 0.037	0.263 ± 0.006
	ECC	0	0.00143 ± 8e−5	0.599 ± 0.049	0.279 ± 0.013
	Ortho. 1	0	**0.00043 ± 3e−5**	0.656 ± 0.038	0.263 ± 0.007
	Ortho. 2	0	0.0014 ± 0.0001	0.636 ± 0.042	0.263 ± 0.019

Logistic regression is used as the base binary classifier

Bold values are the best score for a given dataset between multi-class methods

Table 3 Total classification time, solution time, uncertainty coefficient and Brier score for seven different datasets using five different coding matrices: 1 versus 1, 1 versus the rest, random, orthogonal with no zeros, and orthogonal with zeros

Dataset	Method	Time (s)	Sol. only (s)	U.C.	Brier score
Pendigits	1 versus 1	1.07 ± 0.14	0.409 ± 0.006	**0.985 ± 0.003**	**0.0319 ± 0.0024**
	1 versus rest	**0.84 ± 0.10**	0.082 ± 0.002	0.981 ± 0.003	0.0361 ± 0.0034
	ECC	3.20 ± 0.86	0.13 ± 0.01	0.975 ± 0.004	0.0412 ± 0.0032
	Ortho. 1	2.13 ± 0.89	**0.013 ± 0.002**	0.979 ± 0.004	0.0382 ± 0.0026
	Ortho. 2	1.17 ± 0.28	0.20 ± 0.01	0.982 ± 0.004	0.0354 ± 0.0034
Sat	1 versus 1	**1.39 ± 0.35**	0.077 ± 0.009	**0.800 ± 0.010**	**0.145 ± 0.003**
	1 versus rest	1.70 ± 0.54	0.028 ± 0.005	0.786 ± 0.009	0.153 ± 0.003
	ECC	3.2 ± 1.6	0.04 ± 0.01	0.787 ± 0.011	0.152 ± 0.004
	Ortho. 1	3.8 ± 1.0	**0.008 ± 0.003**	0.792 ± 0.011	0.149 ± 0.003
	Ortho. 2	1.79 ± 0.52	0.034 ± 0.007	0.789 ± 0.009	0.150 ± 0.004
Segment	1 versus 1	0.18 ± 0.05	0.034 ± 0.001	0.923 ± 0.007	**0.0882 ± 0.0053**
	1 versus rest	**0.11 ± 0.03**	0.0102 ± 0.0005	0.919 ± 0.007	0.0938 ± 0.0051
	ECC	0.13 ± 0.07	0.014 ± 0.001	0.915 ± 0.013	0.0938 ± 0.0071
	Ortho. 1	0.16 ± 0.07	**0.0018 ± 0.0001**	**0.925 ± 0.008**	0.0890 ± 0.0048
	Ortho. 2	0.11 ± 0.03	0.015 ± 0.001	0.919 ± 0.012	0.0883 ± 0.0050
Shuttle	1 versus 1	6.3 ± 1.0	0.98 ± 0.06	**0.982 ± 0.003**	**0.0182 ± 0.0015**
	1 versus rest	**6.0 ± 1.6**	0.26 ± 0.03	0.978 ± 0.006	0.0215 ± 0.001
	ECC	12.4 ± 5.7	0.43 ± 0.10	0.878 ± 0.210	0.0731 ± 0.100
	Ortho. 1	10.0 ± 4.7	**0.09 ± 0.03**	0.974 ± 0.003	0.0222 ± 0.0010
	Ortho. 2	6.6 ± 1.6	0.40 ± 0.04	0.978 ± 0.002	0.0230 ± 0.0068
Urban	1 versus 1	0.41 ± 0.21	0.222 ± 0.003	**0.726 ± 0.035**	**0.170 ± 0.009**
	1 versus rest	0.26 ± 0.10	0.0059 ± 7e−4	0.708 ± 0.038	0.176 ± 0.011
	ECC	0.71 ± 0.31	0.0085 ± 0.0011	0.711 ± 0.030	0.178 ± 0.009
	Ortho. 1	0.79 ± 0.24	**0.0014 ± 3e−4**	0.723 ± 0.023	0.173 ± 0.009
	Ortho. 2	**0.22 ± 0.15**	0.0088 ± 0.0011	0.715 ± 0.026	0.172 ± 0.009
Usps	1 versus 1	33.9 ± 17.0	0.42 ± 0.02	**0.929 ± 0.006**	**0.0664 ± 0.0023**
	1 versus rest	**22.9 ± 7.6**	0.110 ± 0.009	0.921 ± 0.005	0.0732 ± 0.0020
	ECC	73.0 ± 29.0	0.150 ± 0.009	0.915 ± 0.006	0.0754 ± 0.0022
	Ortho. 1	70.1 ± 29.0	**0.018 ± 0.003**	0.922 ± 0.006	0.0712 ± 0.0018
	Ortho. 2	34.8 ± 16.0	0.21 ± 0.02	0.920 ± 0.008	0.0707 ± 0.0027
Vehicle	1 versus 1	0.047 ± 0.013	0.00465 ± 8e−5	0.635 ± 0.023	**0.272 ± 0.007**
	1 versus rest	0.055 ± 0.016	0.0016 ± 0.001	0.625 ± 0.033	0.277 ± 0.009
	ECC	0.053 ± 0.024	0.0017 ± 0.0002	0.610 ± 0.061	0.282 ± 0.011
	Ortho. 1	0.050 ± 0.018	**0.00050 ± 3e−5**	0.621 ± 0.032	0.277 ± 0.009
	Ortho. 2	**0.042 ± 0.006**	0.00155 ± 9e−5	**0.639 ± 0.025**	0.278 ± 0.009

A support vector machine is used as the base binary classifier

Bold values are the best score for a given dataset between multi-class methods

trained and applied using the framework provided within libAGF [3, 34, 35]

Results are shown in Tables 2, 3, and 4. Confidence limits represent standard deviations over 10 trials using different, randomly chosen coding matrices. For each trial, datasets were randomly separated into 70% training and 30% test. "U.C" stands for uncertainty coefficient, a skill score based on Shannon's channel capacity that has many advantage over simple fraction of correct guesses or "accuracy" [34, 36, 37]. Probabilities are validated with the Brier score which is root-mean-square error measured against the truth of the class as a 0 or 1 value [16, 38].

For all of the datasets tested, orthogonal ECCs provide a small but significant improvement over random ECCs in both classification accuracy and in the accuracy of the conditional probabilities. This is in line with the literature as in [9, 14]. Improvements range from 0.4% to 17.5% relative (0.004 to 0.139 absolute) in uncertainty coefficient and 0.7% to 10.7% in Brier score. Results are also more consistent for the orthogonal ECCs as given by the calculated error bars.

Also as expected, solution times are extremely fast for the first type of orthogonal ECC. In many cases the times are an order-of-magnitude better than the next fastest method. Depending on the problem and classification

Table 4 Solution time, uncertainty coefficient and Brier score for seven different datasets using five different coding matrices: 1 versus 1, 1 versus the rest, random, orthogonal with no zeros, and orthogonal with zeros

Dataset	Method	Time (s)	Sol. only (s)	U.C.	Brier score
Pendigits	1 versus 1	1.71 ± 0.08	0.45 ± 0.02	**0.977 ± 0.005**	**0.0383 ± 0.003**
	1 versus rest	**0.62 ± 0.02**	0.088 ± 0.004	0.967 ± 0.006	0.0539 ± 0.0021
	ECC	0.77 ± 0.02	0.14 ± 0.01	0.955 ± 0.011	0.0603 ± 0.0061
	Ortho. 1	0.64 ± 0.01	**0.0122 ± 0.0005**	0.961 ± 0.006	0.0560 ± 0.0037
	Ortho. 2	1.3 ± 0.1	0.21 ± 0.02	0.969 ± 0.007	0.0471 ± 0.0033
Sat	1 versus 1	1.97 ± 0.07	0.098 ± 0.02	**0.783 ± 0.009**	**0.159 ± 0.005**
	1 versus rest	**1.17 ± 0.03**	0.035 ± 0.007	0.768 ± 0.012	0.168 ± 0.003
	ECC	1.54 ± 0.05	0.045 ± 0.01	0.765 ± 0.013	0.165 ± 0.004
	Ortho. 1	1.50 ± 0.04	**0.010 ± 0.004**	0.776 ± 0.009	0.162 ± 0.004
	Ortho. 2	1.6 ± 0.2	0.047 ± 0.01	0.763 ± 0.009	0.169 ± 0.010
Segment	1 versus 1	0.170 ± 0.005	0.0353 ± 4e−4	**0.911 ± 0.011**	**0.096 ± 0.005**
	1 versus rest	**0.099 ± 0.0032**	0.0104 ± 4e−4	0.883 ± 0.019	0.119 ± 0.004
	ECC	0.113 ± 0.005	0.015 ± 0.001	0.888 ± 0.026	0.116 ± 0.010
	Ortho. 1	**0.099 ± 0.003**	**0.00190 ± 5e−5**	0.896 ± 0.011	0.115 ± 0.005
	Ortho. 2	0.15 ± 0.01	0.0160 ± 7e−4	0.910 ± 0.011	0.103 ± 0.007
Shuttle	1 versus 1	4.398 ± 0.093	0.90 ± 0.03	**0.981 ± 0.010**	0.0274 ± 0.0110
	1 versus rest	**2.51 ± 0.04**	0.217 ± 0.006	0.967 ± 0.028	0.0315 ± 0.0083
	ECC	2.89 ± 0.06	0.28 ± 0.02	0.972 ± 0.005	0.0313 ± 0.0044
	Ortho. 1	2.63 ± 0.04	**0.045 ± 0.001**	0.976 ± 0.002	**0.0261 ± 0.0010**
	Ortho. 2	3.7 ± 0.3	0.35 ± 0.03	0.976 ± 0.004	0.0270 ± 0.0043
Urban	1 versus 1	0.94 ± 0.02	0.023 ± 0.001	**0.724 ± 0.019**	**0.172 ± 0.009**
	1 versus rest	**0.23 ± 0.01**	0.005 ± 0.001	0.698 ± 0.032	0.184 ± 0.011
	ECC	0.314 ± 0.008	0.008 ± 0.001	0.692 ± 0.028	0.184 ± 0.006
	Ortho. 1	0.31 ± 0.01	**0.0012 ± 4e−4**	0.717 ± 0.022	0.176 ± 0.008
	Ortho. 2	0.44 ± 0.03	0.011 ± 0.001	0.719 ± 0.034	0.176 ± 0.015
Usps	1 versus 1	14.4 ± 0.2	0.41 ± 0.02	**0.914 ± 0.005**	**0.075 ± 0.002**
	1 versus rest	**6.2 ± 0.1**	0.08 ± 0.01	0.897 ± 0.007	0.101 ± 0.002
	ECC	7.5 ± 0.1	0.14 ± 0.02	0.881 ± 0.006	0.095 ± 0.003
	Ortho. 1	7.3 ± 0.1	**0.014 ± 0.004**	0.897 ± 0.006	0.089 ± 0.002
	Ortho. 2	12 ± 1	0.20 ± 0.02	0.899 ± 0.008	0.084 ± 0.003
Vehicle	1 versus 1	0.017 ± 0.005	0.0044 ± 1e−4	**0.628 ± 0.038**	**0.273 ± 0.007**
	1 versus rest	0.017 ± 0.005	0.00156 ± 8e−5	0.607 ± 0.036	0.282 ± 0.007
	ECC	0.02 ± 2.9e−06	0.00158 ± 5e−5	0.602 ± 0.067	0.283 ± 0.014
	Ortho. 1	**0.015 ± 0.005**	**0.00046 ± 1e−5**	0.614 ± 0.026	0.281 ± 0.007
	Ortho. 2	0.016 ± 0.005	0.0015 ± 1e−4	0.597 ± 0.041	0.287 ± 0.011

A piecewise linear classifier is used as the base binary classifier

Bold values are the best score for a given dataset between multi-class methods

method, this may or may not be significant. Since SVM is a relatively slow classifier, solution times are a minor portion of the total. For the logistic regression classifier, solving the constrained optimization problem for the probabilities typically comprises the bulk of classification times. Oddly, the solver for the 1 versus 1 method is the slowest by a wide margin, even though it's a simple (unconstrained) linear solver [31]. This could potentially be improved by using a faster solver [37] or by employing the iterative method given in [31].

The two types of orthogonal ECCs were quite close in accuracy, with sometimes one taking the lead and sometimes the other. For the linear classifier, the second type was always more accurate while the first type was faster. Since it admits zeros, the decision boundaries are usually simpler–see below. For both the SVM and the piecewise linear classifier, skill scores were very similar, differing by at most 2.9% relative, 0.018 absolute, in U.C. and 17% in Brier score. For the SVM, the second type was faster while for the piecewise linear classifier, the first type was faster. The explanation for this follows.

Unfortunately, there is one method that is consistently more accurate than the orthogonal ECCs and this is 1 versus 1. The orthogonal ECCs only beat 1 versus 1 three times out of 21 for the uncertainty coefficient and one time out of 21 for the Brier score. Improvements in uncertainty

coefficient range from insignificant to 0.6% relative or 0.004 absolute. The Brier score improved by 2.6%. Losses using linear classifiers were the worst, peaking at 14.6% relative, 0.203 absolute, in uncertainty coefficient and 50% in Brier score. The results for logistic regression provide a vivid demonstration as to why 1 versus 1 works so well: because it partitions the classes into "least-divisible units", there are fewer training samples provided to each binary classifier, the decision boundary is simpler and a simpler classifier will work better

Nonetheless, there is a potential use case for our method. Although orthogonal ECCs are less accurate than 1 versus 1, they don't lose much. If they are also faster, then a speed improvement may be worth a small hit in accuracy for some applications [3]. While 1 versus 1 beats orthogonal ECCs by a healthy margin using linear classifiers, the biggest loss in U.C. for SVM is only 1.5% relative, 0.011 absolute. Losses for Brier score are somewhat worse, peaking at 6.5%. Unfortunately, because the speed of a multi-class SVM is proportional mainly to the total number of support vectors [3], orthogonal ECCs rarely provide much of a speed advantage. What is needed is a constant-time–ideally very fast–non-linear classifier. This is where the piecewise-linear classifier comes in.

For uncertainty coefficient, 1 versus 1 was always better than orthogonal ECCs when using the piecewise-linear classifier. Losses peak at 1.9 % relative, 0.017 absolute. For the Brier score, only one of the seven datasets showed an improvement over 1 versus 1 at 4.9 %. The worst loss was 39 %. Improvements in speed range from 1.1 % to over 100 %. Much of the speed difference is simply the result of using fewer binary classifiers.

The purpose of the piecewise linear classifier is to improve the speed of the SVM. This speed increase is better with orthogonal ECCs than with 1 versus 1. Orthogonal ECCs applied to piecewise linear classifiers are faster than the the fastest SVM for five out of the seven datasets. Speed often trades off from accuracy. [3] provides a procedure for determining whether it's worth switching algorithms or not. A similar analysis will not be repeated here due to time and space considerations, however whether any improvement in speed is worth the consequent hit in accuracy will depend on the application.

5 Conclusions

As predicted by recent literature, solving for multi-class using orthogonal ECCs was more accurate than the equivalent problem using random ECCs. Unfortunately, they were still unable to beat one-versus-one as an effective multi-class method. The author's own work suggests that the 1 versus 1 classification almost always works well regardless of the dataset [35]. Hsu and Lin [8] find that 1 versus 1 outperform both 1 versus rest and random ECCs on a test of ten different datasets using SVM. One-versus-one is also used, often exclusively, with many statistical classification software packages.

There may still be room for further work, however, with the most likely fruitful line of inquiry being, first, on adaptive methods that use the data to figure out how best to go from binary to multi-class. In [35], for instance, even though 1 versus 1 was almost always most accurate, there was one dataset that benefitted from a more customized treatment. Recent work has focused on both empirically-designed decision trees [5, 6, 39] as well as empirically-designed ECCs [10–12]. Decision trees are the easiest to tackle because there are fewer possibilities and because a tree can be built from either the top down or the bottom up.

A second potential area for future work is in multi-class methods integrated with the base binary classifier, for instance with all the binary classifiers being trained simultaneously [8]. It stands to reason that more integrated multi-class methods would tend to be more accurate than those, such the ones disussed in this note, that treat the binary classifier as a "black box", since there can now be sharing of information.

There is also a potential use case for orthogonal ECCs. If they are paired with a fast, non-linear binary classifier with better than $O(N)$ performance, where N is the number of training samples, orthogonal ECCs should almost always be faster than 1 versus 1 while giving up little in accuracy. The algorithm presented here that solves for the probabilities is simple and elegant and may suggest new directions in the search for more efficient and accurate multi-class classification algorithms. Since it is fast it could help provide speed improvements for such applications as real-time computer vision, image processing, and voice-recognition.

Compliance with ethical standards

Conflict of interest The author declares that he has no conflict of interest.

References

1. Michie D, Spiegelhalter DJ, Tayler CC (1994) Machine learning, neural and statistical classification. Ellis Horwood series in artificial intelligence. Prentice Hall, Upper Saddle River
2. Herman GT, Yeung KTD (1992) On piecewise-linear classification. IEEE Trans Pattern Anal Mach Intell 14(7):782
3. Mills P (2018) Solving for multi-class: a survey and synthesis. Real-Time Image Process. https://doi.org/10.1007/s11554-018-0769-9
4. Müller KR, Mika S, Rätsch G, Tsuda K, Schölkopf B (2001) An introduction to kernel-based learning algorithms. IEEE Trans Neural Networks 12(2):181
5. Cheong S, Oh SH, Lee SY (2004) Support vector machine with binary tree architecture for multi-class classification. Neural Inf Process 2(3):47
6. Lee JS, Oh IS (2003) In: Proceedings of the seventh international conference on document analysis and recognition (IEEE Computer Society). vol 2, pp. 770–774
7. Platt JC, Cristianini N, Shaw-Taylor J (2000) Large margin DAGs for multiclass classification. In: Solla S, Leen T, Mueller KR (eds) Advances in information processing, vol 12. MIT Press, Cambridge, pp 547–553
8. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415
9. Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 2:263
10. Crammer K, Singer Y (2002) On the learnability and design of output codes for multiclass problems. Mach Learn 47(2–3):201
11. Zhou J, Peng H, Suen CY (2008) Data-driven decomposition for multi-class classification. Pattern Recogn 41:67
12. Zhong G, Cheriet M (2013) Binary stochastic representations for large multi-class classification. In: Proceedings of the twenty-third international joint conference on artificial intelligence (IJCAI), pp 1932–1938
13. Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1:113
14. Windeatt T, Ghaderi R (2002) Coding and decoding strategies for multi-class learning problems. Inf Fusion 4(1):11
15. Zhou JT, Tsang IW, Ho SS, Mueller KR (2019) N-ary decomposition for multi-class classification. Mach Learn. https://doi.org/10.1007/s10994-019-05786-2
16. Jolliffe IT, Stephenson DB (2003) Forecast verification: a practitioner's guide in atmospheric science. Wiley, Hoboken
17. Platt J (1999) Advances in large margin classifiers. MIT Press, Cambridge
18. Kong EB, Dietterich TG (1997) Probability estimation via error-correcting output coding. In: International conference on artificial intelligence and soft computing
19. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York
20. Hedayat AS, Sloane NJA, Stufken J (1999) Orthogonal arrays: theory and applications. Springer series in statistics. Springer, New York, chapter 4, pp 61–68
21. Panse MS, Mesham S, Chaware D, Raut A (2014) Error detection using orthogonal code. IOSR J Eng 4(3):2278
22. Arora S, Barak B (2009) Computational complexity: a modern approach. Cambridge University Press, Cambridge
23. Sylvester JJ (1867) Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tesselated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers. Phil Mag 34:461
24. Hedayat A, Wallis W (1978) Hadamard matrices and their applications. Ann Stat 6(6):1184
25. Lawson CL, Hanson RJ (1995) Solving least squares problems, classics in applied mathematics, vol 15. Society for Industrial and Applied Mathematics, Philadelphia
26. Alimoglu F (1996) Combining multiple classifiers for pen-based handwritten digit recognition. Master's thesis, Bogazici University
27. Hull JJ (1994) A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 16(5):550
28. King RD, Feng C, Sutherland A (1995) Statlog: comparision of classification problems on large real-world problems. Appl Artif Intell 9(3):289
29. Johnson B (2013) High resolution urban land cover classification using a competititive multi-scale object-based approach. Remote Sens Lett 4(2):131
30. Siebert J (1987) Vehicle recognition using rule-based methods. Turing Institute, Glasgow
31. Wu TF, Lin CJ, Weng RC (2004) Probability estimates for multi-class classification by pairwise coupling. J Mach Learn Res 5:975
32. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871
33. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27:1
34. Mills P (2011) Efficient statistical classification of satellite measurements. Int J Remote Sens 32(21):6109
35. Mills P (2018) Solving for multi-class: a survey and synthesis. Tech Rep. http://arxiv.org/abs/1809.05929
36. Shannon CE, Weaver W (1963) The mathematical theory of communication. University of Illinois Press, Champaign
37. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C, 2nd edn. Cambridge University Press, Cambridge
38. Brier GW (1950) Verification of forecasts expressed in terms of probability. Mon Weather Rev 78(1):1
39. Benabdeslem K, Bennani Y (2006) Dendrogram-based SVM for multi-class classification. J Comput Inf Technol 14(4):283