Output Coding and Modularity for Multi-Class Problems

André Pastoors
Tom Heskes
Department of Medical Physics and Biophysics, University of Nijmegen
RWCP* Novel Functions SNN† Laboratory
Nijmegen, The Netherlands

Abstract

We study backpropagation networks learning classification problems with multiple classes $k > 3$. The common way to code the output of a network is the one-per-class (OPC) method, where one bit is assigned to each class. A technique called error-correcting output coding (ECOC) converts the k-class learning problem into a large number of two-class learning problems. We propose to use modular architectures as a way to decorrelate the (redundant) network outputs. Various modular architectures are tested on an artificial problem. We conclude that ECOC only improves upon OPC when combined with a sufficiently modular approach.

1 Introduction

In information theory it is customary to add some redundancy to the code vectors to be transmitted in order to be able to reconstruct the original vector, even if some bits are unknown or distorted. This can be achieved by adding some bits to the original code to enlarge the Hamming distance d between the resulting code vectors. It can be shown that the maximal number of errors that can be corrected is $\lfloor \frac{d-1}{2} \rfloor$. The one-per class method (OPC), with as many outputs as classes, yields a highly redundant code: it uses k instead of the minimal ${}^{k}\log 2$ outputs, with k the number of classes. However, the Hamming distance between the code vectors is 2 and thus it is impossible to correct an incorrect bit using the correct bits. An alternative is a technique called error-correcting output coding (ECOC). Code vectors for the ECOC method are even more redundant, but are separated by a larger Hamming distance. For each number of classes $k > 3$ there exists an optimal set of code vectors of length $2^{k-1} - 1$. ECOC can also be viewed as some kind of voting scheme where each bit represents a different vote and yields a different hypothesis. The ECOC strategy works best if the different hypotheses are as independent as possible.

Both the OPC and the ECOC method can be applied to neural classifiers such as multi-layered perceptrons trained with backpropagation. Simulation studies have shown that on some problems the ECOC strategy yields significantly better results [2]. However, in these cases the networks with ECOC

*Real World Computing Partnership
†Foundation for Neural Networks

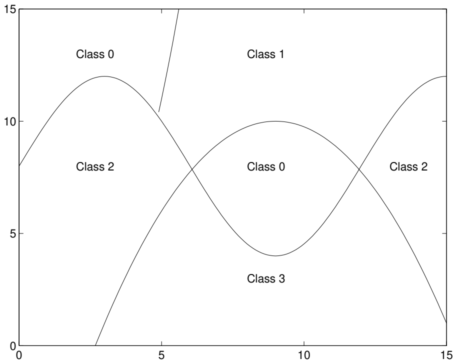

Figure 1: Classification problem.

Class	OPC bits				extra bits		
0	1	0	0	0	1	1	1
1	0	1	0	0	1	0	0
2	0	0	1	0	0	1	0
3	0	0	0	1	0	0	1

Table 1: Error-correcting output code.

required a large amount of hidden units and computation time, probably to decorrelate the votes of the different output bits. An easier way to avoid correlations is to use a modular approach, e.g. by using separate networks for each output bit. For quite different reasons such a modular approach has been suggested in combination with OPC [1]. In this paper we will compare different codings and architectures on an artificial problem with $k = 4$ classes.

2 Simulation paradigm

Our artificial learning problem, taken from [4], is sketched in figure 1. There are two continuously valued inputs and 4 classes. Tabel 1 shows the optimal error-correcting output code for a 4-class problem: the first 4 bits identical to the one-per-class code with 3 additional bits representing class combinations. Network outputs are assigned to the class with the closest code vector according the Hamming distance $d(\vec{o}, \vec{c}_i) \equiv \sum_{j=1}^{N} |o_j - c_{ij}|$, where $\vec{o}$ denotes the network output, $\vec{c}_i$ the code vector for class i, and N the number of outputs. We compare 7 different architectures: not only ordinary OPC, ordinary ECOC, modular OPC, and modular ECOC (top row in figure 2), but also architectures with an intermediate level of modularity: four modules, five modules, and two blocks (bottom row). These architectures differ in the extent and the way they try to decorrelate the various bits.

In our simulations a multi-layered perceptron is trained using a conjugate-gradient implementation of backpropagation. Each architecture is trained on 50 training sets of 60 patterns, randomly drawn from the four-class distribution sketched in figure 1. To prevent overfitting, training is stopped when the performance on a large validation set of 540 randomly drawn patterns seriously starts to deteriorate. All modules are trained and stopped separately. The optimal number of hidden units is determined through cross-validation and appears to be around 5 hidden units per output for blocks of hidden units connected to several outputs and 8 hidden units for modules connected to just one output. The generalization performance of each run is measured on account of the entire grid $[0, 15] \times [0, 15]$ with a resolution of 0.2.

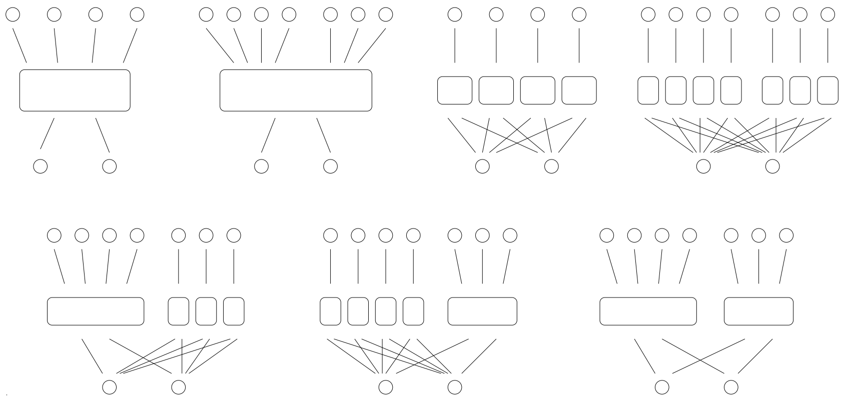

Figure 2: The various architectures compared in our simulations. Each oval represents an independent block of hidden units.

3 Discussion of results

The results of our simulations are summarized in table 2 where the various architectures are ranked according to their percentage of correct classifications on the independent test set. Two methods are called significantly different if in at least 35 of the 50 runs one method outperforms the other (binomial test with $p = 0.01$). Our results imply that ECOC yields a significant improvement over standard OPC if combined with a modular architecture. This modular architecture should be chosen such that the extra output bits are as independent as possible, both from the bits representing OPC and from each other. With less modularity ECOC no longer outperforms OPC, more modularity does not help but does not harm either.

The hidden units of a multi-layered perceptron build hyperplanes to separate the different classes. With all outputs connected to one block of hidden units, the extra output units will have a strong tendency to use the same hyperplanes as the first three units. The high correlation between the extra bits and the OPC bits makes it difficult to correct errors as well as to spoil correct classifications. This is easily confirmed by our simulations where the extra bits in the ordinary ECOC architecture correct and spoil only 0.08% and 0.06%, respectively, of the classifications made solely on account of the first four bits.

It is much less clear why it helps to give each extra output bit its own module. Extra bits that share their hidden units ("two blocks" and "five modules") correct about 2.5% and spoil about 1.5% of the classifications suggested by the OPC bits. ECOC architectures with separate modules for the extra bits ("four modules" and "modular ECOC") still spoil about 1.5%, yet correct about 4.0% of the OPC suggestions. No such an improvement in performance is seen when the first four bits have their own separate module. This apparent difference between the first four bits and the last three bits may be explained by the fact that for a four-class problem the code representing OPC is even more redundant than the code assigned to the last three bits.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
(1) four modules	0	0	+	+	+	+	+
(2) modular ECOC	0	0	+	+	+	+	+
(3) two blocks	-	-	0	0	0	0	+
(4) five modules	-	-	0	0	0	0	+
(5) ordinary ECOC	-	-	0	0	0	0	0
(6) ordinary OPC	-	-	0	0	0	0	0
(7) modular OPC	-	-	-	-	0	0	0

Table 2: Ranking and significance of differences between the architectures.

4 Conclusion and further directions

The higher the number of classes, the more one can win by using a highly redundant error-correcting output code instead of the standard one-per class code. We have shown that even on a problem with only 4 classes ECOC outperforms OPC, provided that it is combined with a modular architecture to decorrelate the extra bits from the OPC bits and from each other. We expect to find that classification problems with many classes will also benefit from a modular approach.

The effect of ECOC may be described in terms of bias and variance [3, 4]. For neural networks, bias and variance are more or less interchangeable, since by increasing the number of hidden units, we can always build a less biased model at the price of a higher variance. ECOC, especially when combined with a modular approach, requires a larger number of network weights than OPC. This may lead to a lower bias, while the increase in variance due to the larger number of weights is compensated by a decrease due to the combination of multiple votes. Further studies should substantiate these arguments.

References

[1] R. Anand, K. Mehrotra, C. Mohan, and S. Ranka. Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks*, 6:117–124, 1995.

[2] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[3] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[4] E. Kong and T. Dietterich. Error-correcting output coding corrects bias and variance. Technical report, Oregon State University, Corvallis, OR, 1995.