ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 113-118

Coding the Outputs of Multilayer Feedforward. [1]

† Mercedes Fernández-Redondo - † Carlos Hernández-Espinosa.

† Universidad Jaume I, Campus de Riu Sec, Edificio TI, Departamento de Informática, 12080 Castellón, Spain. e-mail: redondo@inf.uji.es, espinosa@inf.uji.es

Abstract. In this paper, we present an empirical comparison among four different schemes of coding the outputs of a Multilayer Feedforward networks. Results are obtained for eight different classification problems from the UCI repository of machine learning databases. Our results show that the usual codification is superior to the rest in the case of using one output unit per class. However, if we use several output units per class we can obtain an improvement in the generalization performance and in this case the noisy codification seems to be more appropriate.

1. Introduction.

One important topic in the design of a neural network is the codification of the information contained in the training set. The training set is the only source of knowledge about the problem used by the neural network in order to solve it. So we can conclude that this information and its codification should be an important issue.
In the case of the inputs, the codification is easy and natural for the case of numerical or ordered inputs. We just use the numerical values as the inputs of the neural network.
However, for the outputs the situation is different. A one of N, 1-of-N, codification is used; we usually assign one output unit to each of the classes. This codification seems logical, but it is rather arbitrary in the sense that many other possibilities can be used.
This may be the reason why other output codification alternatives were proposed recently in the bibliography.
For example in [3] it is proposed a target adaptation for the output; the targets are adapted for the case of the most difficult to learn patterns.
In [2] it is proposed to train the neural network by adding gaussian additive noise to the target.
In the reference [1], it is proposed to use multiple redundant outputs as an alternative to train an ensemble of neural networks to improve the generalization performance.
Finally, in references [4], [5], [6] it is proposed a kind of self-organization in order to obtain the targets of the network. The targets are obtained as a result of the learning process. They support their research by arguing that the task of assigning targets to the networks has no biological plausibility. Unfortunately, the experimental results of the paper show a lower generalization capability with respect to the usual codification. Because of this reason, the practical interest of the self-codification is limited.

[1] Work funded by a project number GV-99-75-1-14 from Generalitat Valenciana.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 113-118

All these papers are recent and there is a lack of results and comparison among the different output codification schemes. We have several alternatives to codify the output and there are no experimental results to know which one is the best.
The objective of this paper is to present a comparison among the most interesting codification schemes.
At this point, we should ask ourselves which factors of the network convergence are affected by the codification of the output. We think that they are the generalization capability, the speed of convergence and the probability of correct convergence. This hypothesis is confirmed by our experimental results.
The organization of the paper is as follows. In the next section we present a review of the output codification methods that will be compared. In section three we present and explain the experimental results. Finally we conclude in section four.

2. Theory.

In this section we will review the different coding schemes that will be compared in the experimental results section.

2.1. Normal.

This is just the usual or normal codification of outputs. It is included in order to provide a reference for comparison.
The codification have one output unit per class and the unit has a target value of one if the sample belong to the class; in this case, the rest of output units have a target value of zero. This codification is also known as 1-of-N.

2.2. Target Adaptation.

This codification was proposed in reference [3]. The number of output units is also the number of classes. However, the target values are adapted during the training process.
The basic idea is to assign a target equal to the output value when the sample is correctly classified and increment the target value in the other case in order to make the error signal more relevant.
The coding scheme can be resumed by the following algorithm:

1) Repeat during the training process:
   2) For sample *k*, calculate the outputs *O* of the network.
   3) If the sample is correctly classified the next target, T_K, of this sample will be the output value of the network, $T_K=O$.
   4) If the sample is not correctly classified the next target, T_K, of this sample is the output value plus 1 in the case of the correct class and the output minus one for the rest, equation (1).

$$T_k = \begin{cases} O_i + 1, & \text{if the correct class is } i \\ O_i - 1, & \text{for the rest of outputs} \end{cases} \qquad \textbf{(1)}$$

2.3. Noise.

In reference [2], it is proposed to add gaussian random noise to the usual 1-of-N codification of the target values. The number of outputs is also equal to the number of classes and the codification is the usual 1-of-N. The difference is that we use an error noise added to the target as in equation (2)

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 113-118

The noise is gaussian distributed with a fix variance of sigma, σ^2. The appropriate value of the variance in the reference seems to be 0.001.

$$T_k(noisy) = T_k(without\ noise) + n \qquad (2)$$

It is also use an annealing schedule during the training process for the learning step as in equation (3). Where *N(t)* is the epoch number of the training process and *c* is a constant which is problem dependent and should be determined by trial and error.

$$\eta(t) = \eta_0 \cdot \frac{1}{1 + \frac{N(t)}{c}} \qquad (3)$$

2.4. Multiple Outputs per class.

In the reference [1] it is proposed to use more than one output unit per class. This idea is an interesting alternative to training an ensemble of networks for solving a problem. If we use N output units per class, the codification of the targets will have N target values of one for the outputs representing the class and the rest of outputs will have a target value of zero.

By analogy to the usual codification scheme, we can call this codification N-of-M. Where N is the number of output units per class and M the total number of outputs.

After training, we will have several outputs representing a class and we should decide how to classify a sample. For this, it is interesting to use the already known combination schemes in the neural network ensemble field. The two most popular schemes of combination are voting and averaging. We have finally decided to average the results because it seems it has a better performance.

So, in order to decide the classification of a sample, we average the outputs representing to the same class and we reduce the N-of-M codification to the usual 1-of-N.

This codification scheme can be used together with the rest and that is what we have done in our experiments.

3. Experimental results.

The main purpose of this research was to experimentally evaluate the different output codification methods and determine their performance. We have applied the four methods to eight different classification problems. They are from the UCI repository of machine learning databases. Their names are Balance Scale (BALANCE), Cylinders Bands (BANDS), Liver Disorders (BUPA), Glass identification (GLASS), Heart Disease (HEART), the Monk's Problems (MONK1, MONK2) and Voting Records (VOTING). The complete data and a full description can be found in the UCI repository.

In order to apply the methods and see their performance we have to train at least one neural network with each method. However, the performance of a neural network depends on the partition of the data in training, test and cross-validation sets and also on the initialization of the weights.

So, in our methodology we performed 30 trials with different partitions of the data and random seeds for every method; the performance values will be mean of the 30 trials.

We evaluated the three main effects of the output coding: the speed of convergence, the generalization performance and the ratio of successfully convergence.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 113-118

The speed of convergence was measured by the epoch of convergence. This epoch was determined by cross-validation.
The second issue, generalization performance, was measured by calculating the percentage correct (PC) with the test set. Finally, the probability of successful convergence was measured by the number of networks which did not converge with respect to the total 30 trials.
The results are in tables 1-4. In these tables we have the performance for each database in each row. In the columns, we have three main columns with the performance of each method and the main columns are divided in other three columns. The first one (header "POR") contains the mean percentage correct (the generalization performance), the second (header "N") contains the number of network that did not converge from the total of 30 trials (the probability of successful convergence). Finally, the third column (header "ITE") contains the convergence iteration (the speed of convergence).
The tables are for different number of output units per class. We have used the values 1, 3, 5 and 8 output units per class.
By observing table 1, we can see that the generalization performance of the normal codification is clearly better than the one of target adaptation. Also, the number of

	NORMAL			TARGET ADAPTATION			NOISE		
	POR	**N**	**ITE**	**POR**	**N**	**ITE**	**POR**	**N**	**ITE**
BALANCE	87.6±0.6	1	1170±190	86.8±1.0	1	2800±300	87.5±0.6	0	2300±400
BAND	72.4±1.0	10	360±120	67.0±1.2	8	180±20	72.6±1.0	10	230±50
BUPA	58.3±0.6	1	1700±400	49.0±1.6	4	2600±400	58.5±0.6	7	6300±800
GLAS	78.7±0.9	2	1630±110	52±2	8	550±60	79.3±0.7	6	4700±300
HEART	82.1±0.9	2	120±40	76.2±1.3	2	280±70	82.3±0.8	4	400±200
MOK1	74.6±	9	1030±140	61±2	11	1120±160	74.5±0.9	3	1700±300
MOK2	66.0±0.5	2	1200±200	54±2	1	720±40	66.1±.5	8	4200±600
VOTING	95.3±0.4	1	70±20	95.1±0.4	0	170±30	95.8±0.4	0	130±90

Table 1. Performance of different coding schemes in the case of one output per class.

	NORMAL			TARGET ADAPTATION			NOISE		
	POR	**N**	**ITE**	**POR**	**N**	**ITE**	**POR**	**N**	**ITE**
BALANCE	88.3±0.6	1	760±180	88.5±0.5	4	2200±300	88.3±0.6	0	1800±400
BAND	72.5±1.0	11	130±20	68.2±1.1	10	96±13	72.5±1.0	8	210±60
BUPA	59.3±0.8	5	1200±300	54.6±1.6	5	3100±500	58±0.6	2	7000±700
GLAS	77.8±0.8	5	1360±120	66±2	11	550±40	78±0.8	4	3500±400
HEART	82.1±0.8	2	300±170	77.2±1.3	2	290±90	81.9±0.8	8	300±200
MOK1	81.6±1.1	11	990±170	73.4±1.4	12	1190±170	81.0±1.1	4	970±160
MOK2	66.2±0.5	4	2300±400	62.6±1.3	1	1100±300	66.3±0.5	3	3900±700
VOTING	95.0±0.3	1	100±40	95.0±0.5	1	64±14	95.3±0.4	0	90±30

Table 2. Performance of different coding schemes in the case of three outputs per class.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 113-118

	NORMAL			TARGET ADAPTATION			NOISE		
	POR	**N**	**ITE**	**POR**	**N**	**ITE**	**POR**	**N**	**ITE**
BALANCE	90.2±0.6	2	1100200±	88.6±0.7	3	2500±300	90.3±0.6	0	1100±200
BAND	71.9±1.0	8	110±20	68.5±1.2	12	101±14	71.9±1.0	5	270±90
BUPA	60.6±0.8	8	800±200	52.7±1.6	6	2400±400	58.8±0.7	4	5900±700
GLAS	78.3±0.8	5	1310±130	71.8±1.9	11	500±50	78.6±0.7	5	3300±400
HEART	81.6±0.8	3	360±150	78.3±1.3	2	61±15	81.8±0.8	4	400±200
MOK1	84.3±1.2	17	830±150	76.5±1.6	8	1140±180	83.7±1.0	3	1400±300
MOK2	66.2±0.5	7	2900±400	60.3±1.6	1	2300±400	66.3±0.5	11	3800±600
VOTING	94.8±0.3	0	150±50	94.7±0.4	1	47±7	95.3±0.3	0	180±110

Table 3. Performance of different coding schemes in the case of five outputs per class.

	NORMAL			TARGET ADAPTATION			NOISE		
	POR	**N**	**ITE**	**POR**	**N**	**ITE**	**POR**	**N**	**ITE**
BALANCE	91.9±0.5	3	630±180	88.8±0.5	5	2200±300	91.6±0.6	0	630±180
BAND	71.9±1.0	8	270±140	68.0±0.9	14	103±14	72.0±1.0	5	390±190
BUPA	62.4±0.7	3	550±170	54.2±1.4	10	2400±400	60.3±0.8	2	5200±700
GLAS	78.9±0.7	1	1060±110	76.1±0.7	13	530±50	79±0.8	3	3800±400
HEART	81.1±0.8	2	350±180	78.1±1.2	3	390±130	81.0±0.8	4	310±170
MOK1	85.9±1.1	16	560±110	77±2	9	910±170	84.5±1.0	6	1000±200
MOK2	66.0±0.6	13	4000±400	63.1±1.3	6	2700±400	66.3±0.5	6	2400±600
VOTING	94.6±0.3	1	90±40	94.5±0.4	1	40±5	95.0±0.3	0	250±130

Table 4. Performance of different coding schemes in the case of eight outputs per class.

converged networks is better.

We can also see that the normal codification has a generalization capability similar to the one of noise codification. However, the number of converged networks and the speed is better for the case of the normal codification.

So, we can conclude that the usual 1-of-N codification may be superior to the rest for the case of using one output unit per class.

By observing tables 1 to 4, we can see that for the databases BALANCE, BUPA and MOK1 we got an improvement in the generalization capability as the number of output units increases. So, the increment in generalization capability is problem dependent. This result is already known in the bibliography of neural network ensembles.

We can also see that in the case of the normal and target adaptation codification the number of networks which did not converge increases as the number of output units increases for some databases. This result makes the use of multiple outputs per class rather limited to a low number. Otherwise, we would obtain a large number of unconverged networks.

However, this last tendency is the opposite for the case of the noise codification. In this case the number of networks that did not converge decreases as the number of output units increases. So, it is very interesting to use this codification together with multiple outputs.
Another interesting results is that the speed of convergence increases as the number of output units increases. So, the computational cost of training with multiple outputs is partially compensated by the speed.

4. Conclusions.

We have presented a comparison among four different schemes of coding the outputs of a Multilayer Feedforward network. The four different schemes are evaluated on a total of eight different classification problems. The speed of convergence, the generalization performance and the probability of successful convergence were measured and compared.
We found that the normal codification may be superior to the rest for the case of using one output unit per class. However, if we use more than output unit per class we can get an improvement in the generalization performance depending on the problem and in this case, the right codification is noisy targets because it has a better probability of successful convergence.

References.

1. Sarkar, D. Improving generalization through multiple redundant output units. Proceedings of World Congress on Neural Networks, vol. 3, pp. 593-597, 1993.
2. Wang, C., Principe, J.C. Training Neural Networks with Additive Noise in the Desired Signal. Proceedings of the International Conference on Neural Network, pp. 1084-1088, 1998.
3. Adeney, K., Korenberg, M. Target Adaptation to improve the performance of least-squares classifiers. Proceedings of the International Conference on Neural Networks, 2000.
4. Al-Mashouq, K. Coding the outputs of a Feedforward Neural Net. Connection Science, vol. 9, no. 2, pp. 217-228, 1997.
5. Sarukkai, R. Solving XOR with a single layered Perceptron by supervised self-organization of multiple output labels per class. Proceedings of the International Conference on Neural Networks, vol. 5, pp. 2807-2810, 1995.
6. Sarukkai, R. Supervised Networks that Self-Organize Class Outputs. Neural Computation, vol. 9, pp. 637-548, 1997.
7. Sarukkai, R. Supervised Self-Coding in Multilayered Feedforward Networks. IEEE Transactions on Neural Neworks, vol. 7, no. 5, pp. 1184-1195, 1996.