Creating Modular-like Ensembles by Output Clustering

I. Mora-Jiménez, A. Lyhyaoui, J. Arenas-García and A.R. Figueiras-Vidal
Department of Signal Theory and Communications
Universidad Carlos III de Madrid
Avda. Universidad 30, 28911 Leganés (Madrid) SPAIN.
{*inmoji, abdel, jarenas, arfv*}*@tsc.uc3m.es*

ABSTRACT

In this paper we consider the possibility of replacing the output layer of Multi-Layer Perceptrons (MLPs) by local schemes when dealing with classification problems. In order to open the possibility of developing LMS-trainable models, and posterior adaptive schemes, we apply a trainable version of the classical k-Nearest Neighbour classifier (kNN) named kNN-Learning Vector Classifier. We develop the corresponding training formulas for the whole resulting structure and apply it to some classification benchmark problems. The experimental results give evidence of the nearly systematic advantage of our proposal with respect to MLPs, as well as of their competitive performance regarding the Modular Neural Networks (MNNs), which have a similar philosophy as our approach.

1 INTRODUCTION

The expressive power of Multi-Layer Perceptrons (MLPs) has been perceived from their very beginning, as the efforts of Widrow to find training algorithms indicate (Widrow and Lehr, 1990). The explosive increase in their analysis and applications obeys to the same perception. Today, many proofs of the universal approximation capacity of MLPs are available, as (Cybenko, 1989), for example. The same can be said of many other Neural Network (NN) architectures, such as Radial Basis Function Networks (RBFNs), that are not global approximators as MLPs, but local fitting schemes. This duality is one of the relevant open problems in NN research: how to combine schemes to take advantage of both global and local approximation capabilities.

The fact of dealing with a structure that has a Single Layer Perceptron (SLP) at the output forces that, to reach an optimal behaviour, the (transformed) data distributions at the input of this layer must be linearly separable. This can be hard to be provided by the previous layer(s). Therefore, it seems that the study of alternative procedures to extract information from the previous layers is a reasonable subject of research.

We can consider that the layer ahead the linear combination in an MLP has the role of extracting characteristics from data in a manner that allows that a (new) output classification scheme can decide with success. An interesting option consists on using a local

decision scheme at the output, because this opens the possibility of combining the global operation mode of the previous layers with this localized way of decision: in principle, a desirable characteristic for any classifier.

To implement the above idea we present a new architecture, the Single Layer Perceptron-Learning Vector Classifier (SLP-LVC). Basically it is a net with two layers: the first one is composed by a number of SLPs, which are combined by a Learning Vector Classifier (LVC) with local properties (Mora-Jiménez et al., 2002). For reasons of clarity we will reduce the discussion here to binary classification (C_0 and C_1 classes).

The rest of the paper is organised as follows. We first review the local classifier from which the SLP-LVC is designed. Next section connects it with the SLPs and derive the corresponding formulas to train our scheme. In Section 4 we show and discuss some experimental work. Finally some concluding remarks are made in Section 5.

2 THE LEARNING VECTOR CLASSIFIER (LVC)

In (Mora-Jiménez et al., 2002) we have already presented the architecture of this classifier, based on a new variant of the kNN rule (Fukunaga, 1990). It uses a reduced set of centroids instead of the whole training dataset, what provides some advantages. First, it avoids the high storage and computational burden of the kNN methods while preserving its localized behaviour. Besides, it allows a trainable method, very useful when combining with other schemes to improve the classifier performance. The only parameters of the kNN-LVC are: the set of labelled centroids $\{\mathbf{c}_0\}$ and $\{\mathbf{c}_1\}$ for classes C_0 and C_1, respectively, and the number k of centroids which are adjusted in the training stage.

So then, the kNN-LVC provides the following classification rule for an n-dimensional sample $\mathbf{x}$

$$\frac{k_1}{||\mathbf{x} - \mathbf{c}_1{}^{(k_1)}||_2^n} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{k_0}{||\mathbf{x} - \mathbf{c}_0{}^{(k_0)}||_2^n} \tag{1}$$

where $||.||_2$ represents the Euclidean distance, k_i is the number of the k nearest centroids to sample $\mathbf{x}$ belonging to class C_i ($k = k_1 + k_0$), and $\mathbf{c}_i{}^{(k_i)}$ is the k_i-th corresponding centroid of class C_i. As it is explained in (Mora-Jiménez et al., 2002), this formulation derives from the Bayes' decision rule for minimum error (Fukunaga, 1990), where the a posteriori probabilities are got through an estimate of the probability density function via the kNN estimator.

Assuming t is the target value for sample $\mathbf{x}$ (1 if $\mathbf{x} \in C_1$, -1 if $\mathbf{x} \in C_0$), we derive from (1) the following discriminant function D, where $D > 0$ if $\mathbf{x}$ is assigned to the correct class

$$D = t\left(\frac{||\mathbf{x} - \mathbf{c}_0{}^{(k_0)}||_2^2}{k_0^{2/n}} - \frac{||\mathbf{x} - \mathbf{c}_1{}^{(k_1)}||_2^2}{k_1^{2/n}}\right) \tag{2}$$

The training of this architecture consists in updating the position of the centroids to maximize D for each pattern $\mathbf{x}$ (stochastic training). It is achieved by means of a gradient algorithm whose formulation will be provided in the next section.

We have decided to use this local classifier because it is little sensitive to local minima in the training process, avoiding thus many of the problems a classifier like the RBFN would

incur. As for other local schemes as Learning Vector Quantizers (LVQs) (Kohonen, 1990), we discard them because although they solve the local minima problem of the RBFNs, they are not suitable to include in schemes where the objective is to maximize the classification rate: LVQs are devised to get a representation of the data space, even though the supervised training allows them to get a good performance. Finally, we have not used the simplest of the local methods, the kNN rule, because it does not provide a trainable scheme.

3 THE WHOLE CLASSIFIER SLP-LVC AND ITS TRAINING

The SLP-LVC is based on the principle of combining the global ($\mathbf{o}$) and local ($\mathbf{x}$) features of a dataset. Figure 1 represents the structure of our SLP-LVC, where the "S/C" (Selector/Combiner) block provides the inputs to the LVC. In the "S" mode it only takes the global or local features, but in the "C" mode it creates an augmented data space.

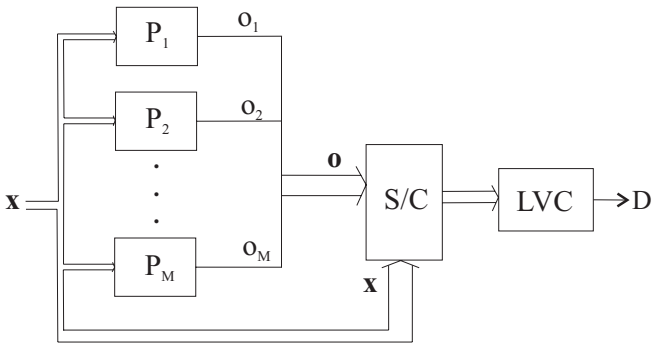

Figure 1: Structure of the SLP-LVC classifier. Both outputs $\mathbf{o}$ of the single layer perceptrons P_m and data $\mathbf{x}$ can feed the Learning Vector Classifier.

This idea is not completely new: the modular architectures that Jordan and Jacobs proposed (Jacobs et al., 1991) work according to similar principles. The experts provide outputs that are selected (by weighting) by a gate circuit (a network that establish "regions" for the input data). So, we are speaking here on an alternative version of this approach, in which the role of the gate element is played by a local classifier. As in Figure 1 is indicated, it can be fed by the samples, and not only by previous layers' outputs.

The general form of the samples entering to the kNN-LVC is $\mathbf{X} = [\mathbf{o}\ \mathbf{x}]$ where $\mathbf{o}$ is a M-dimensional vector with components o_m coming from the outputs of the SLPs

$$o_m = \tanh\left(\sum_{l=1}^{n} w_{ml} x_l + w_{m0}\right) \tag{3}$$

and w_{ml}, w_{m0}, are the weights of the m-th perceptron.

From eqs. (2) and (3), it is immediate to write the gradient formulas for the weights, where N is the dimension of the space entering the LVC: it can be $N = m$ if only the SLP outputs are considered, $N = n$ if we take the original data or $N = m+n$ for the combination

mode in the "S/C" block

$$\frac{\partial D}{\partial w_{ml}} = \frac{\partial D}{\partial o_m}\frac{\partial o_m}{\partial w_{ml}} \tag{4}$$

$$\frac{\partial D}{\partial o_m} = 2t\left(\frac{o_m - c_{0,m}^{(k_0)}}{k_0^{2/N}} - \frac{o_m - c_{1,m}^{(k_1)}}{k_1^{2/N}}\right) \tag{5}$$

$$\frac{\partial o_m}{\partial w_{ml}} = (1 - o_m^2)x_l, \quad (x_l = 1 \text{ if } l = 0) \tag{6}$$

Regarding the training of the LVC, the gradients of D with respect to the k_1-th and k_0-th centroids are

$$\nabla_{\mathbf{c}_1^{(k_1)}} D = \frac{2t}{k_1^{2/N}}(\mathbf{X} - \mathbf{c}_1^{(k_1)}) \tag{7a}$$

$$\nabla_{\mathbf{c}_0^{(k_0)}} D = -\frac{2t}{k_0^{2/N}}(\mathbf{X} - \mathbf{c}_0^{(k_0)}) \tag{7b}$$

Note that in (5) we are only considering the position of the k_0-th and k_1-th centroids. Besides, if k_0 or k_1 is equal to zero for a datum, it does not contribute to the perceptron training, although it can be used by the local classifier. In practice, we train all the k nearest centroids to a sample.

It should be mentioned that (6) is, as for the standard MLPs, likely responsible for adaptability limitations. However, in our scheme, the use of sigmoids for the single layer perceptrons is not an essential ingredient: it can be removed, and we continue dealing with a scheme in which the linear combinations can contribute to the adequate shaping of the data before the LVC.

For an effective training of the SLP-LVC we must take into account that the modification of the SLP coefficients (w_{ml}) changes the distribution of data to which the local classifier is applied. So, even though the weights variation due to one sample can be small, the modification produced by all training samples may vary significantly the distribution presented to the LVC, making the centroids' position is not adequate. So, we have decided to implement an epoch-by-epoch training process in two steps. The first one corresponds to the SLP training (in batch mode) and the second is the adjusting of the centroids location (stochastic mode). However, before carrying out the last step, it is possible that the starting centroids are a bad representation of the new dataset, therefore it is necessary to reposition them. To do that, for each centroid we determine the two nearest training data in the transformed space and identify their position in the observation space. In this way we create a synthetic centroid (average of the two samples) in the input space that, after passing through the SLPs with updated weights, determine the new centroid in the N-space. Finally we adjust the centroids using eqs. (7a) and (7b). This process is repeated until a minimum error in a validation set is achieved or we reach a fixed number of epochs.

4 RESULTS

In this section we will check the performance of the proposed SLP-LVC by comparing it to other two perceptron based schemes. The first one corresponds to the classical two-layer MLP, trained to minimize a quadratic error objective. The second one is the MNN

ensemble proposed by (Jacobs et al., 1991), where the purpose is to use ensembles of NNs to decompose the problem into simpler ones than can easily be solved by classifiers called experts. Their combination is controlled by a gating network, having the role of defining regions for the input samples.

The training of the MNN is usually carried out by minimizing a cost funtion E with stochastic steepest gradient descent algorithm. There are different choices for E, but we have used the one proposed by (Jacobs et al., 1991)

$$E(\mathbf{x}) = -ln \sum_{i=1}^{M} g_i(\mathbf{x}) \exp\left(-\frac{1}{2}(t - o_i(\mathbf{x}))^2\right) \tag{8}$$

In our implementation we have used SLPs with and without the hyperbolic function for all the experts and a MLP network (with no hidden layers) for the gating circuit (coefficients g_i in (8)), whose M outputs are activated with a softmax function.

We have tested these approaches on four datasets. The first two are synthetic and bi-dimensional problems: "ripley" and "kwok", having a Bayessian classification limit of 92% and 88.7% respectively. The two real datasets are from the UCI Machine Learning Repository[1]: "sonar" ($n = 60$) and "pima diabetes" ($n = 8$). As for the latter there is no test set we have created 10 partitions with the 40% of the whole set. All the data have been normalized to interval $[-1, 1]$. The results we will show correspond to the average classification rate on the test set, obtained through 10 runs with different initial conditions for all the parameters. In all cases the training is stopped according to an epoch-by-epoch cross-validation procedure applied to the probability error, using 20% of the training samples (randomly selected in each trial).

As for the the training conditions, we have made experiments with three values for the number of SLPs ($M = 2, 4$ and 6). The SLP weights are initialized for all the methods with values from a uniform distribution with range $[-0.1, 0.1]$. For the MLP we have applied an adaptive learning rate (with 0.01 as starting value), in order to get better results. For the MNN we have explored five initial values for the learning rate (the same for experts and gate): $\mu_i = [1, 0.8, 0.6, 0.4, 0.2]$ with a decreasing schedule for their evolution (in inverse proportion to the epoch number). Regarding the SLP-LVC, we have used two different learning rates for each layer (maintained constant along the training): the corresponding to the LVC is set to 0.05 and for the SLPs we examine the values $[0.001, 0.005, 0.01, 0.05, 0.1]$. In our approach we have also considered different initial conditions for the SLP weights, obtained stopping the training of the MLP after a maximum number of epochs: $0, 5, 10$ and 20, since the SLP outputs will be more suitable to be separated with an hyperplane as the training proceeds. For the LVC part we consider $10, 15$ and 20 centroids per class and $k = 2, 3, 4, 5$. In Table 1 we show the correct classification percentages achieved according to the maximum rate in cross-validation when the SLP-LVC works in the "S" mode, just considering the outputs of the SLPs.

Note that even MLP results are reasonable, the scheme we propose improve them in all cases excepting the real problems with few experts ($M = 2$). The reason for this behaviour

[1] Available at http://www.ics.uci.edu/~mlearn/MLRepository.html

	Ripley			Kwok			Diabetes			Sonar		
	2	4	6	2	4	6	2	4	6	2	4	6
MLP	88.3	87.8	88.0	84.3	86.6	84.1	77.6	76.3	76.1	80.7	81.6	78.4
MNN	87.8	88.0	88.4	87.7	87.4	87.8	75.9	75.8	76.7	82.0	84.7	85.4
SLP-LVC	89.7	89.9	89.5	87.9	87.6	87.7	76.2	76.7	77.0	77.8	82.1	80.1

Table 1: Average classification accuracy (in %) for different structures, problems (first row) and number of SLPs (second row) when the LVC is fed with the SLP outputs.

can be the high dimension of the observation space in relation to the number of SLPs considered. The MNN seems to be stucked in local minima during the training when few experts are considered (see "ripley" and "diabetes"), although it usually outperforms MLP, specially when the number of experts increase. Comparing SLP-LVC with MNN we can check that we offer a competitive performance in most cases, even though these experiments correspond to the SLP-LVC where the local classifier is just fed with the SLP outputs. Because of brevity we have not shown the results corresponding to the combination of both global and local features, although they will be presented at the conference.

5 Conclusions

The possibility of replacing the output linear combiner of binary MLP classifiers for a (less restrictive) local decision scheme has been explored. In order to allow LMS-type training, and subsequent adaptivity, we have used a local scheme named kNN-LVC. Experimental results of SLP-LVC for several typical classification datasets show that it outperforms MLPs in almost all the cases and that it offers a very competitive performance even compared with schemes based on the mixture of experts.

References

Cybenko, G. (1989). Approximation by Superposition of a Sigmoidal Function. *Math. Control Signal Systems*, 2:303–314.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* New York, NY: Academic Press, 2nd edn., 1990.

Jacobs R. A., Jordan M. I., Nowlan, S. J. and Hinton, G. E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87.

Kohonen, T. (1990). The Self-Organizing MAP. *Proc. IEEE*, 78:1464–1480.

Mora-Jiménez, I., Lyhyaoui, A., Arenas-García, J., Navia-Vázquez, A., and Figueiras-Vidal, A. R. (2002). A Trainable Classifier via k Nearest Neighbors. *Accepted for the HIS'02.*

Widrow, B. and Lehr, M. A. (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madalines, and Backpropagation. *Proc. IEEE*, 78:1415–1444.