7 Local Stabilisation of Ensembles of LVQ-based Nearest Neighbour Classifiers

Abstract- This chapter presents two simple methods that generate a single classifier from an ensemble of Nearest Neighbour (NN) classifiers. These methods are different to any combination method for classification proposed to date. They are based on getting an ensemble of perturbed NN classifiers using a Kohonen's LVQ algorithm that is initialised with different ordering of training data and different initial conditions. Since the distribution of the codebooks of the NN ensemble is arranged in clusters, a *combined* codebook is computed using a cluster algorithm, like the batch K-means, that computes an average solution in each natural group. This first method (so-called *local averaging*) can be viewed as either a technique to reduce the variance of codevectors or as the result of averaging a series of particular bootstrap replicates. The second method (*local extreme*) is based on the idea of generating a single codebook from the extreme prototypes of the whole set of codebooks. A simple heuristic algorithm is proposed in order to get these extreme prototypes. Although the variance of *local extreme* respect to the codebooks of the ensemble is greater than the variance of *local averaging*, it allows combining codebooks of different sizes and also can lead to better solutions in problems where several solutions co-exist. Experimental results using seven classification problems confirm the utility of both methods.

Index Terms- Combination of Classifiers, Ensemble learning, Averaging, Nearest Neighbour Classifiers, Kohonen's LVQ algorithms.

1. Introduction

Ensemble learning has emerged in recent years as a technique to improve the generalization performance of those systems that can learn from examples. The idea is to combine an independent collection of learning systems (or predictors) that have been all trained in the same task. This combination is typically done by majority in classification (eg. (Breiman, 1994)) or by averaging in regression (e.g. (Breiman, 1994)(Perrone, 1993)), although more complex forms like mixture of experts (e.g. (Jacobs et al., 1991)) or boosting (e.g. (Schapire, 1999)) are possible as well. In order to gain something from using an ensemble, the single predictors that form it must be different in order to produce a better-combined solution. This can be simply achieved using different training sets but it is impractical in real-world cases since data usually is scarce. In this case, a certain level of independence can be attained forming from a single set a collection of training sets with techniques like resampling (e.g. bootstrapping (Efron & Tibshirani, 1994)), focussing (e.g. stacked generalization (Wolpert, 1992), boosting (Schapire, 1999), (Jacobs et al., 1991)) or hybrid procedures (e.g. half- &-half bagging (Breiman, 1998), (Avnimelech & Intrator, 1999)). Another possible way of getting some degree of variability between the predictors of the ensemble is to vary several parameters like predictor's architecture, input noise injection, training times, initial conditions and ordering of the training data (in on-line learning) (see (Natftaly et al., 1997)(Hansen & Salamon, 1990)(Partridge & Yates, 1996)).

This chapter presents two methods that generates a combined version of an ensemble of NN classifiers that have been trained using a Kohonen's LVQ algorithm (Kohonen, 1996) initialized with different ordering of training data and different initial conditions. They are based on (1) the local nature of NN classifiers and (2) the form of the distribution of the perturbed codebooks of the NN ensemble that is arranged in clusters. The first method (*local averaging*) achieves a single codebook using the batch K-means that computes a centroid in each natural group. The interpretation of these centroids is simple: they form a codebook to which the execution of the learning algorithm tends in average. The second method (*local extreme*) relies on the idea of generating a single codebook from the extreme prototypes of the whole set of codebooks. A simple heuristic algorithm is proposed in order to get these extreme prototypes. While local averaging is less sensitive to the particular ensemble of codebooks used for computing the combined codebook, local extreme allows combining codebooks of different sizes in a natural way and also can lead to better solutions in problems where several solutions co-exist. Both

methods tend to stabilize class borders so classification accuracy could be improved in some cases, as we will show in our work.

In the next section, the two methods for combining NN's codebooks are presented. Section 3 introduces a theoretical analysis to understand how local averaging works and why can improve in some cases the classification accuracy. In section 4, experimental results are presented. Finally, some discussion and conclusions are given.

2. Local Stabilisation

The use of different initial conditions in learning systems usually leads to different predictors since the optimisation method typically searches a local minimum of its cost function so it is possible that using different initial conditions the system stacks at a different local minimum points. Besides, if we employ a validation set to stop training, the learning system can reach the same minimum of the validation cost function at different points of the training cost function. Lastly, in the case of on-line algorithms, the ordering of the training data in a cyclic sampling modifies these minimum (or fixed) points (e.g. LVQ1 (Bermejo, 2000a)). Hence, due to three distinct sources of variability, we can obtain a certain degree of difference between predictors that we will easily interpret in the case of NN classifiers trained with LVQ algorithms. This explanation will lead us to introduce our stabilisation methods.

NN classifiers assigns an input pattern $\mathbf{x} \in \Re^p$ to a class label from a finite set $\Im=\{1, ..., c\}$ using this local function:

$$\mathrm{NN}(\mathbf{x}) = \sum_{j=1}^{K} 1(\mathbf{x} \in R_j)\,\text{class label}\,(\mathbf{m}_j) \tag{1}$$

where the function class label($\mathbf{x}$) returns the class label associated to pattern $\mathbf{x}$, $\mathbf{1}$(condition) is the indicator function which is 1 if condition is true and 0 otherwise, $\{\mathbf{m_j}, j=1,...,K\}$ are the codevectors that form the classifier's codebook $\mathbf{C}$ and R_j is the region of influence (or Voronoi region) associated to the codevector $\mathbf{m_j}$ defined by

$$R_j = \left\{ \mathbf{x} \,\middle|\, \|\mathbf{x}-\mathbf{m}_j\| = \min_{i=1,..,K} \|\mathbf{x}-\mathbf{m}_i\| \right\} \tag{2}$$

Given two different initial codebooks of size K, named $\mathbf{C}^a[0]$ y $\mathbf{C}^b[0]$, for a LVQ algorithm that uses a cyclic sampling of a different ordering of the training set and the early stopping

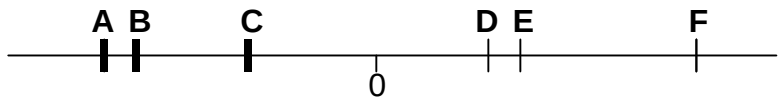
technique, one could anticipate that the distance between the resulting codebooks $\mathbf{C}^a$ and $\mathbf{C}^b$ defined by

$$d(\mathbf{C}^a, \mathbf{C}^b) = \sum_{i=1}^{K} \left\| \mathbf{m}_i^a - \arg \min_{\mathbf{m}_j^b} \min_{j=1,\ldots,K} \left\| \mathbf{m}_i^a - \mathbf{m}_j^b \right\| \right\|^2 \quad (3)$$

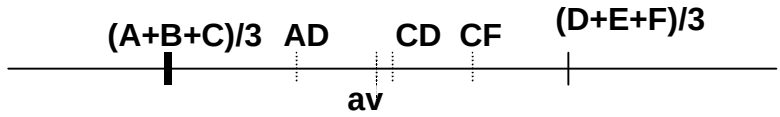
will be small. This can be expected since, at least, the LVQ1 and LVQ2 algorithms are on-line gradient descent algorithms (see (Bermejo, 2000a)(Bottou, 1998)). Therefore, two systems trained with the above conditions achieve a similar value of its associated cost function so the two solutions might resemble. More specifically, the LVQ1 algorithm is a design algorithm for a vector quantizer (VQ) (Gersho & Gray, 1992) that discretely approximates a modified probability density function (p.d.f.) defined in the input space (Bermejo, 2000a)(LaVigna, 1990). In consequence, since the quantifier is a local function based on Voronoi regions, the two resulting quantifiers have similar Voronoi regions so $\mathbf{C}^a$ and $\mathbf{C}^b$ must resemble. Thus, if we obtain a collection of codebooks using the above strategy, the overall set of codevectors might be grouped in clusters. Due to the computed codevectors are estimators of a certain class of local expectations, it makes sense to obtain locally (e.g. in each cluster) an averaging since, at this level, the problem for stabilising these kind of estimators is the same than more general settlings like regression. However other forms of local stabilisation could arise if we focus on stabilising explicitly the class borders of the NN classifier instead.

Suppose that we have to discriminate among 1-D patterns belonging to class A and B where this problem can be solved with a 1-NN classifier of 2 prototypes. Figure 1a shows the (possible) computed prototypes in 3 different training sessions. While the frontier points AD=(A+D)/2 and CF=(C+F)/2 in figure 1b mark the extreme frontiers, the point CD, formed with the extreme prototypes of each class, falls between the extreme frontier and consequently stabilise in some degree the set of possible solutions. Note that this point can notably vary if we compute it from another training sessions. On the other hand, the point computed with local averaging av(ABCDEF)=((A+B+C)/3+(D+E+F)/3)/2 (that also falls inside the extreme frontiers so it helps to stabilise the set of solutions) uses the whole set of prototypes to form the estimator so it is less dependent on their particular values as the number of training sessions augments. However *local averaging* can be degraded in the presence of some outliers (e.g. some prototypes that achieve a worse solution and are far from the others) while the second proposed method (*local extreme*) could be beneficial in these cases since they are not affected by this kind of prototypes. While in the example of figure 1 we could detect poor solutions (since e.g. they cause a high classification error) and remove them from local averaging, this is not easy in a

general case (K>2). Since the learning algorithm in a single training session could lead to poor solutions in some regions and good solutions in others, it remains very difficult to determine what prototypes must be deleted.



a)



b)

Fig.1. a) A possible set of prototypes computed in 3 training sessions.(Prototypes of class A are designed with a thick line while prototypes belonging to class B are labelled with a thin line.); b) Frontier points computed with local averaging (av) and local extreme (CD). (Frontier points are denoted with a discontinuous line.)

2.1. The Local Averaging Algorithm

According to the above considerations, we propose the following method for generating a combined version of an ensemble of nearest neighbour classifiers based on local averaging:

1. Execute Q times a Kohonen's LVQ algorithm that is started with different initial conditions of a codebook (of size K) and different position of the training data in the memory array.
2. Compute K cluster centroids over the totality of the computed codevectors (QxK) in the training sessions using the batch K-means with the restriction that the algorithm only will average codevectors of the same class. The execution of the K-means is stopped when the validation error of a NN classifier that uses the resulting averaged codebook (the K current centroids) increases.

Consequently, our first proposed method can be seen as the local application of the averaging technique used in regression. The most remarkable difference between our method and typical averaging (or any other combination technique) is that the combined NN classifier does not

employ the ensemble of NN classifiers anymore. Instead, it has a single codebook (of the same size than the members of the ensemble) that is formed with the K centroids.

2.2. The Local Extreme Algorithm

Once we obtain Q codebooks $\{\mathbf{C}_i, i=1,...,Q\}$ we only must retain the extreme prototype of each class. In fact this extreme prototypes are easily detected in the fussed codebook $\mathbf{C}_F = \bigcup_{i=1}^{Q} \mathbf{C}_i$ since the other prototypes can be deleted without affecting the classification accuracy. Hence the algorithm to detect these extreme prototypes can be summarised as follows:

1. Execute Q times a Kohonen's LVQ algorithm that is started with different initial conditions of a codebook (of size K) and different position of the training data in the memory array.
2. Compute the fussed codebook $\mathbf{C}_F = \bigcup_{i=1}^{Q} \mathbf{C}_i$ with the Q codebooks $\{\mathbf{C}_i, i=1,...,Q\}$ of the training sessions. Then, remove those codevectors in which no training data are assigned to them (pass 1). Finally, delete those codevectors that classification accuracy of the validation set stands or improves if they are not used in the NN classifier, (pass 2). This second step is repeated cyclically through the codebook until there is no 'redundant' codevector left.

3. Theoretical Analysis of Local Averaging.

3.1. Local averaging as a variance reduction technique.

It is well known that averaging is used to reduce the variance part of the bias/variance decomposition of the generalization error (Geman et. al, 1992) in regression problems (e.g.(Natftaly et al., 1997)). As we will show in this section, our method does it too. We can see this variance reduction if we interpret the computation's goal LVQ algorithms as to approximate discretely the following p.d.f.:

$$f_Y(\mathbf{y}) = \sum_{i=1}^{C} f_{Y|B_i}(\mathbf{y}) P(B_i); \quad f_{Y|B_i}(\mathbf{x}) = K_i \left(f_{X|C_i}(\mathbf{x}) P(C_i) - f_{X|C_j}(\mathbf{x}) P(C_j) \right) \tag{4}$$

where B_i denotes the event of belonging to the Bayes region of class i, $\left\{ f_{Y|B_i}(\mathbf{y}), i=1,...,C \right\}$ are the density functions conditioned to belong to each Bayes region, $\{P(B_i), i=1,...,C\}$ are the probability of belonging to these regions, K_i is a constant that ensures

$$\int f_{Y|B_i}(\mathbf{y})d\mathbf{y} = 1, \tag{5}$$

$$f_{X|Ci}P(C_i) = \max_{r=1..C,\forall \vec{x}\in Bi} f_{X|C_r}(\mathbf{x})P(C_r) \tag{6}$$

$$f_{X|Cj}P(C_j) = \max_{r\neq i,\forall \vec{x}\in Bi} f_{X|Cr}(\mathbf{x})P(C_r) \tag{7}$$

with $\left\{f_{X|C_r}(\mathbf{x}), r = 1,...,C\right\}$ are the class p.d.f.'s and $P(C_r)$ are the class priors (see (Kohonen,1996) for additional information).

Thus, the optimal codevectors are simply the conditioned expectations $E_{Y\backslash Ri}[\mathbf{y}\backslash R_i]$ (with R_i as the associated Voronoi region) like in standard VQ (Gersho & Gray, 1992). Then, we can measure the generalization error of the system as the expected quantization error in Y defined as:

$$\begin{aligned} L &= E_{DY}\left[\left\|\mathbf{y} - VQ(\mathbf{y})\right\|^2\right] = L_{approximtion} + L_{estimation} = \\ &= E_X\left[\left\|\mathbf{y} - VQ_{opt}(\mathbf{y})\right\|^2\right] + E_{DY}\left[\left\|VQ_{opt}(\mathbf{y}) - VQ(\mathbf{y})\right\|^2\right] \end{aligned} \tag{7}$$

where E_{DY} denotes the expected operator over Y and all possible data sets D of size N and Vopt is defined by

$$VQ_{opt}(\mathbf{y}) = \sum_{i=1}^{K} 1(\mathbf{y}\in R_i)E_{Y|R_i}(\mathbf{y}|R_i) \tag{8}$$

with

$$R_i = \left\{\mathbf{y} \;\middle|\; \left\|\mathbf{y} - E_{Y|R_i}(\mathbf{y}|R_i)\right\| = \min_{j=1..K}\left\|\mathbf{y} - E_{Y|R_j}(\mathbf{y}|R_j)\right\|\right\} \tag{9}$$

and VQ is determined by

$$VQ(\mathbf{y}) = \sum_{i=1}^{K} 1(\mathbf{y}\in \hat{R}_i)\mathbf{m}_i \text{ with } \mathbf{m}_i = \hat{E}_{Y|\hat{R}_i}(\mathbf{y}|\hat{R}_i) \tag{10}$$

If the estimation error is low (e.g. $\hat{R}_i \to R_i$) then it can be approximated by

$$L_{estimation} = \sum_{i=1}^{K} L_{estimation}\left|R_i\right. P(R_i) = \sum_{i=1}^{K} E_D E_{Y|Ri}\left[\left\| E_{Y|Ri}(\mathbf{y}|R_i) - \mathbf{m}_i \right\|^2\right] P(R_i) \tag{11}$$

The estimation error in each region R_i admits the bias/variance decomposition, yielding

$$L_{estimation}\left|R_i\right. = \text{bias}^2(\mathbf{m}_i) + \text{var}(\mathbf{m}_i) = \left\| E_{Y|Ri}(\mathbf{y}|R_i) - E_D(\mathbf{m}_i) \right\|^2 + E_D\left(\left\| \mathbf{m}_i - E_D(\mathbf{m}_i) \right\|^2\right) \tag{12}$$

Suppose now that we obtain an averaged codebook using the method introduced in the last section. Thus, each resulting codevector $\mathbf{m_i}$ is the empirical mean of an ensemble of Q codevectors $\{\mathbf{m_i}^j, j=1,...,Q\}$ which all approximate $E_{Y|R_i}(\mathbf{y}|R_i)$. Accordingly we can write down $\mathbf{m_i}$ as

$$\mathbf{m}_i = \frac{1}{Q}\sum_{j=1}^{Q} \mathbf{m}_i^j \tag{13}$$

Then, the variance part of the estimation error in R_i of $\mathbf{m_i}$ gives (see a similar result in (Natftaly et al., 1997))

$$\text{var}(\mathbf{m}_i) = E_D\left(\left\| \mathbf{m}_i - E_D(\mathbf{m}_i) \right\|^2\right) = Q^{-1}\sum_{j=1}^{Q} E_D\left(\left(\mathbf{m}_i^j - E_D(\mathbf{m}_i^j)\right)^T (\mathbf{m}_i - E_D(\mathbf{m}_i))\right) = Q^{-1}\sum_{j=1}^{Q} \text{cov ar}_D(\mathbf{m}_i^j, \mathbf{m}_i) \tag{14}$$

Consequently, var($\mathbf{m_i}$) is decreased as each codevector of the ensemble $\mathbf{m_i}^j$ is less correlated with $\mathbf{m_i}$. Following (Natftaly et al., 1997), the variance of each component of $\mathbf{m_i}$ is bounded by the following expression

$$Q^{-1} \min_{j=1...Q} \text{var}(m_{ik}^j) \le \text{var}(m_{ik}) \le \max_{j=1...Q} \text{var}(m_{ik}^j) \quad k = 1,..., p \tag{15}$$

Therefore, the reduction of var($\mathbf{m_i}$) goes typically (e.g. when var($\mathbf{m_{ik}}^j$) is approximately equal for all j=1,...,Q) from a factor 1/Q (in the case of minimum correlation) to produce no improvement at all. But, since estimation is only important to classification purposes near class boundaries, we can expect that only the variance reduction of the codevectors which form class boundaries will cause an improvement of the classification accuracy.

3.2. Local averaging as a combination of bootstrap replicates.

Suppose that, to generate the ensemble, we use the LVQ1 algorithm with a constant step size α and a cyclic sampling of training data. In concordance with (Bermejo, 2000a), the LVQ1 approximately converges to these K fixed points

$$\mathbf{m}_i = \sum_{s=0}^{N_i-1} f[s]\,\mathrm{sign}\left(\mathrm{class\,label}(\mathbf{x}_i[s]) = \mathrm{class\,label}(\mathbf{m}_i)\right)\mathbf{x}_i[s], \qquad i = 1,\ldots,K \tag{16}$$

where $\mathbf{x_i}[k]$ is those training data that fall in R_i ordered in the way that are located in memory, f[k] is a function that depends on the sequence $\{\mathbf{x_i}[k]\ ,\ s=0,\ldots,N_i-1\}$ and sign(u) is 1 if u is true and -1 otherwise. If we generate a new training set from a bootstrap replicate (Efron & Tibshirani, 1994) of the original training set and then perform gradient descent over the LVQ1 loss function without the optimization errors of the LVQ1 algorithm (e.g. using the Newton optimization method (Bermejo, 2000a)), the fixed points would follow the general expression given in equation 8. It is well known that the average of a series of bootstrap statistics converges to the statistic computed from the original training set, as the number of bootstrap replicates tends to be large enough (Efron & Tibshirani, 1994). Thus, our method of local averaging can be understood as the averaging of a 'bootstrap' replicates originated due to the dynamics of on-line learning that is particularly affected by the ordering of the training data in the memory array. Then, as Q increases, the averaged codebook converges to that codebook which the execution of the learning algorithm tends in mean. For the other LVQ algorithms, a similar conclusion could be derived since the essential point here is that on-line learning is affected by the ordering of training data in memory.

4. Experimental Results

In this section, we present simulations of the local averaging method in seven classification problems. To get some insight into how this method works, we include three artificial problems, one which involves only two classes that have been generated from a mixture of non-radial gaussians (fig. 2a), the other taken from (Ripley,1994) and finally a 1-D problem with two gaussian classes. We also incorporate four real databases: the Finnish Speech database (Kohonen et al., 1995), the Satimage database (Murphy & Aha, 1994) and the NIST hand-written databases for uppercase and lowercase characters (Garris, 1997). The Finish speech database (Kohonen et al., 1995) contains 3964 cepstral-coefficient vectors picked up from continuos Finnish speech, from the same speaker. Each vector has a

dimension of 20 and has been labelled to represent one of the 20 possible phonemes. The Satimage database was taken originally from the UCI Repository of machine learning databases (Murphy & Aha, 1994). The Australian Centre for Remote Sensing generated it from Landsat Multi-Spectral Scanner image data from NASA. This database contains 6435 patterns with 36 components belonging to five classes. Upper & Lower hand-written databases can be found in (Garris et al., 1997). These images (32x32 pixels) belonging to 26 classes have been preprocessed with a Principal Component Analyzer that extracts 64 components from each image (NIST's mis2evt utility). The correlation matrix of the PCA was computed using the training set. We have split the original databases in training, validation and test sets (table 1) (except in the case of Ripley's problem).

4.1. Experiment #1: Local Averaging

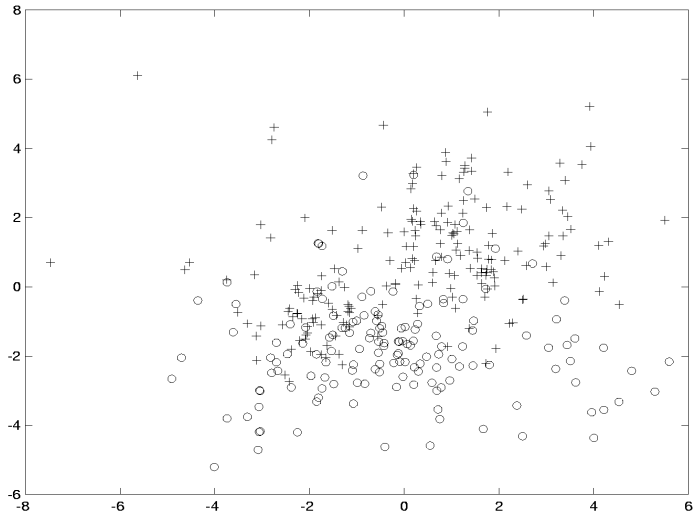
We test local averaging using our 2-D artificial problem and the Satimage and Finish databases. In all three problems, we use the LVQ1 algorithm with constant step size and cyclic sampling of training data. The initial values of the codebooks are been computed with the LVQ_PAK's eveninit program (Kohonen et al., 1995). LVQ1's parameters, K (the size of each codebook) and optimal training time have been estimated using the validation set. We have generated the averaged codebook for several sizes (Q) of the ensemble where the initial value of the batch K-means is one of the codebook of the ensemble that was randomly chosen. Then, we compare the results of the averaged NN classifier with the best NN classifier selected from the ensemble using the validation set.

Figure 2b shows the overall computed codevectors (QxK) in an ensemble with Q=100 and K=8.As expected the distribution of these codevectors is agglutinated in K clusters. In figure 2c, we can see the averaged codebook computed with our averaging method, which effectively tends to compute cluster's centroids. Note that the averaging of 6 out of 8 clusters affects the classification accuracy that corresponds to those codevectors that the NN classifier use to form class boundaries. We display, in figures 2d, 3 and 4, the test and validation errors of the best NN classifier from the ensemble and the NN classifier that employ the averaged codebook for several values of Q and the 2D, Finnish Speech and Satimage problems respectively. In the first two problems, for any value of Q, the averaged NN classifier outperforms the best one from the ensemble. The improvement in the classification rate goes from 0.5% to 1.5%. On the other hand, our method only is slightly superior to the best NN classifier when Q=100 in the Satimage problem. As figure 5 shows the best classifier of the

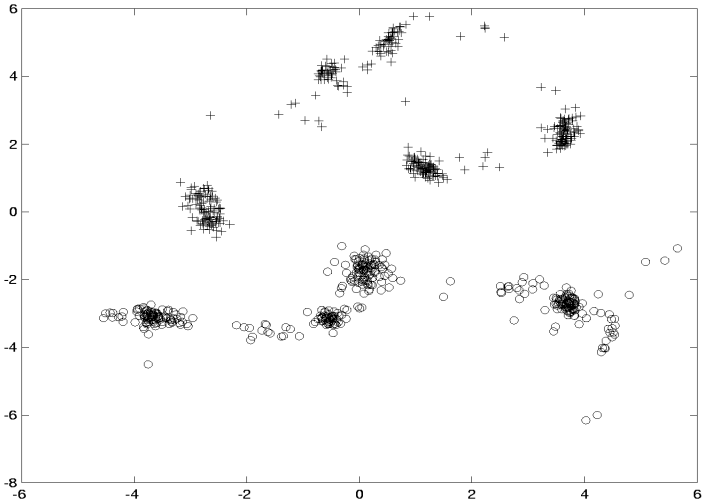
ensemble remains only superior to the best-averaged classifier in 3 out of 30 cases. Our methods achieves a relative improvement that goes from 0,44 % to 17,85 %.

Database	*Training Set Size*	*Validation Set Size*	*Test Set Size*
2D Problem	1000	1000	35000
Ripley	250	-	1000
Speech	1962	647	1315
Satimage	4293	712	1430
Upper	24420	1031	10453
Lower	24205	1078	10914

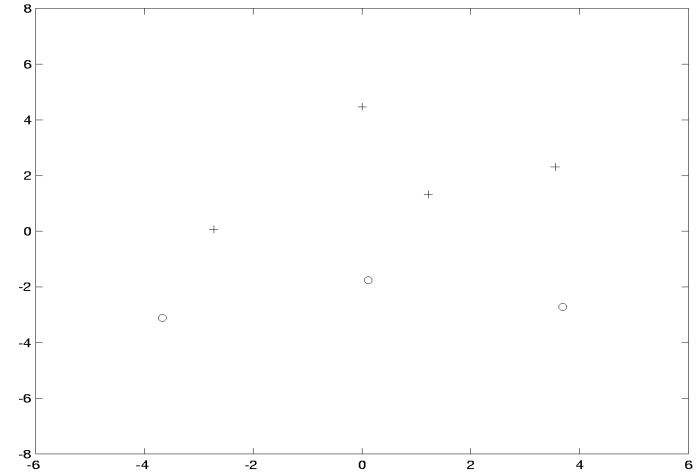
Table 1. Size of databases.



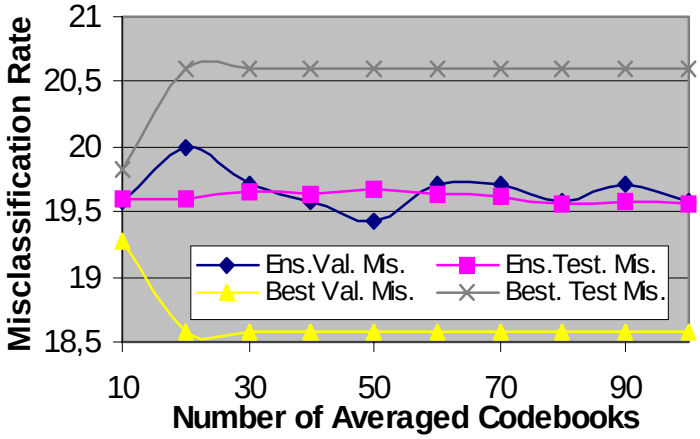
a)



b)



c)



d)

Fig. 2. Classification Results of the 2D Problem: a) the 2D Problem, b) distribution of the codevectors of a ensemble of size 100, c) the averaged codebook of b), d) the misclassification rate vs. the number of averaged codebooks (we display the validation and test errors of the averaged and best NN classifiers)

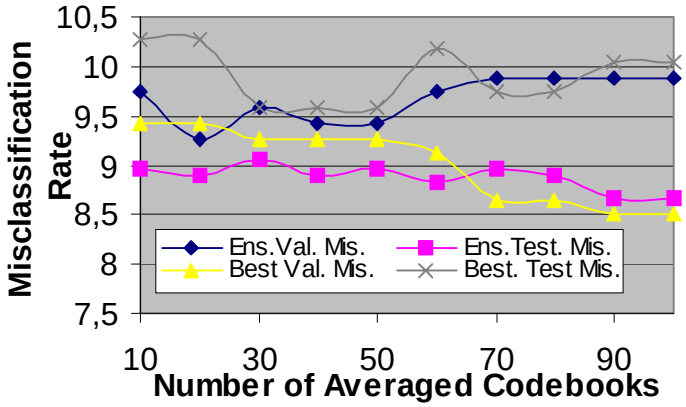


Fig. 3. Misclassification rate vs. the number of averaged codebooks in the Speech problem.

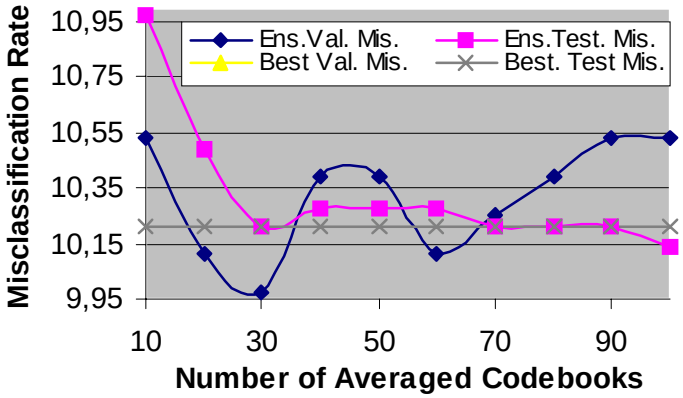


Fig. 4. Misclassification rate vs. the number of averaged codebooks in the Satimage problem.

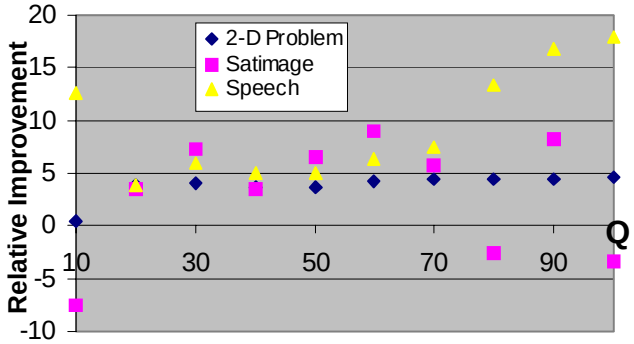


Fig.5. Relative improvement (%) of local averaging for the test set.

4.2. Experiment #2: Reduced Local Extreme

We have generated ten medium-sized codebook for the upper & lower handwritten recognition problems using the DLVQ3 algorithm (Bermejo, 2000b). Then we apply steps 1 and 2.1 of *local extreme*. We compare the final NN classifier computed with the reduced version of local extreme with a NN classifier that uses the whole training points and a set of prototypes (of the same size of the 'local extreme' set) computed with an execution in cascade of Kohonen's LVQ algorithm. First we initialize with *eveninit* and *balance* and then we execute *olvq1*, *lvq1*, *lvq2* and *lvq3* programs (see (Kohonen et. al, 1995) for further details). The optimal parameters of all these algorithms have been estimated with the validation sets.

As we can see in table 2, local extreme achieves a recognition accuracy on test set of 94.14% in uppercase handwritten recognition, while the others 1-NN classifiers only give 93.27% (using the whole training set) and 92.03% (on average for the static LVQ). In lowercase handwritten recognition, the best classifier is the static LVQ-based NN classifier (86.26%) closely followed by local extreme (86.22%). However the use of the reduced local extreme allows computing a codebook of large size of similar accuracy using less training time since the training time of

LVQ grows exponentially as the codebook size grows. Thus, the training of ten codebooks of 300 prototypes and the reduced version of local extreme is much less time-demanding than training with 3000 prototypes.

Algorithm	Upper				Lower			
	Tra.	Val.	Test	**C** size	Tra.	Val.	Test	**C** size
DLVQ3	2.7/0.2†	7.4/.4	7.2/.1	611/25	12.1/.2	15.4/.3	15.2/.1	296/9.1
Local Extreme (Pass 1)	**1.19**	**6.31**	**5.86**	**4354**	10.59	14.16	13.74	2405
Static LVQ	4.1/0.3	8.7/.5	7.9/.2	4398	8.5/.45	14.2/.4	**13.7**/.2	2448
1-NN	-	5.84	6.73	24420	-	14.55	14.07	24205

Table 2. Results of the experiment #2. We show for each classifier no. of cycles ran, training, validation and test error and codebook size. († Computed mean/variance over ten runs)

4.3. Experiment #3: Local Averaging vs. Local Extreme

4.3.1. 2D Problem

We have generated 100 codebooks $\{\mathbf{C}_i, i=1,...,100\}$ for the 2D problem with the DLVQ1 algorithm (Bermejo, 2000b). The optimal parameters of DLVQ1 have been estimated with the validation set. Then we apply *local extreme* and *local averaging*. The codebooks have different sizes since DLVQ1 is a dynamic learning algorithm that incrementally grows and deletes the number of prototypes so depending on the particular conditions of each learning sessions DLVQ1 reach different solutions. Consequently, the number of clusters in local averaging is determined by the maximum size of the ensemble of codebooks.

Figure 6 shows how local extreme works. The fussed codebook $\mathbf{C}_F = \bigcup_{i=1}^{100} \mathbf{C}_i$ stabilises in some degree (top right) the differences in the solutions that appear in the training sessions (top left). The application of local extreme simply gives a condensed codebook that tends to retain the same solution than the fussed codebook. Since the heuristic algorithm removes those codevectors that do not cause a decrease in the classification accuracy of the validation, consequently we expect that these prototypes do not contribute to form class borders and the final codebook can give a similar solution than the fused codebook. As one can observe in figure 6, pass 1 of the heuristic algorithm deletes a considerable number of prototypes in a single pass through training data. Hence, the time to compute the extreme prototypes in pass 2 is reduced. On the other hand, local averaging detects the eight main clusters and computes their centroids (fig.7). While both methods improve the classification accuracy respect to the best classifier of the ensemble (figures 8 and 9), local extreme achieve a greater relative improvement (fig. 10).

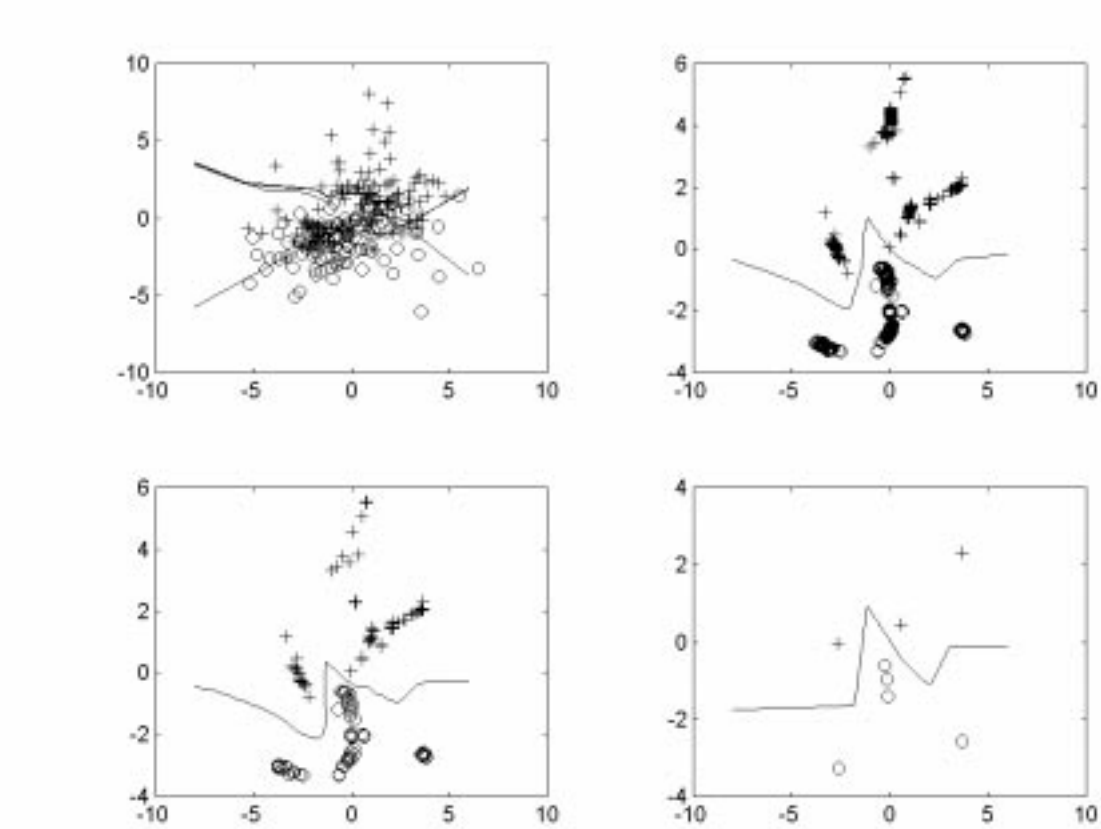


Fig.6. Local extreme for the 2D Problem: top left) training data and several solutions computed with DLVQ1 (note that different solutions co-exist); top right) the fussed codebook of 100 training sessions and the NN class border using this codebook; bottom left) the remaining codebook once we apply step 2.1 of local extreme and the class border for these prototypes; bottom right) The final codebook of local extreme and its class border.

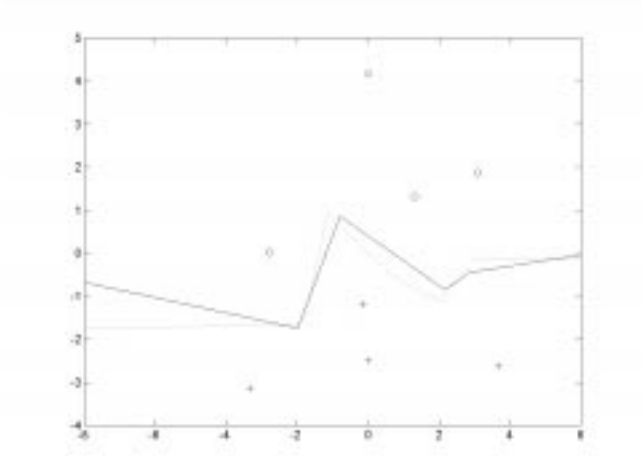


Fig.7. Local extreme (dotted line) vs. local averaging (continuos line) for the 2D Problem. We also show the set of prototypes computed with local averaging.

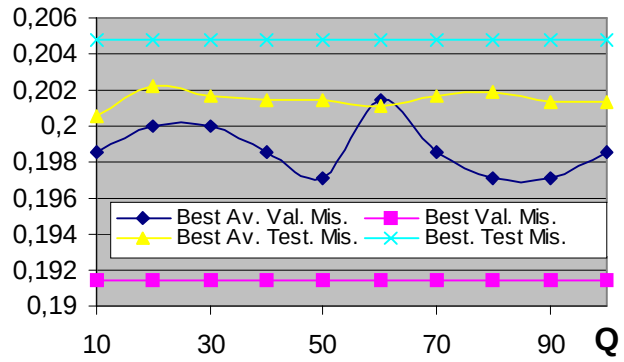


Fig. 8. Misclassification rate vs. the number of averaged codebooks (Q) in the 2D problem using DLVQ1.

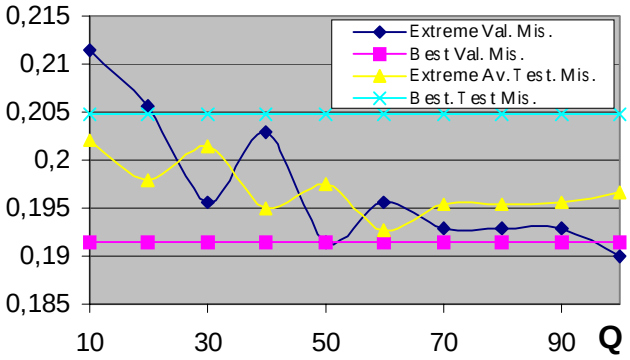


Fig. 9. Misclassification rate for local extreme vs. Q in the 2D problem using DLVQ1.

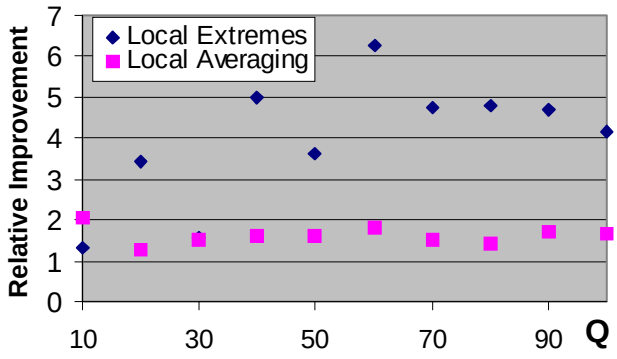


Fig.10. Relative improvement (%) of local averaging and local extreme in the 2D problem using DLVQ1 for the test set.

4.3.2. Ripley's Problem

We generate again 100 codebooks using LVQ1 with 4 prototypes trained for Ripley's Problem (Ripley, 1994). Since the solutions in the ensemble are very similar (fig.11), these methods do not improve the classification accuracy. Both converge to an (almost) identical solution (figs.11 and 12) with an error of 8.9%. However the best solution of the ensemble gives 8.8 % (fig.13).

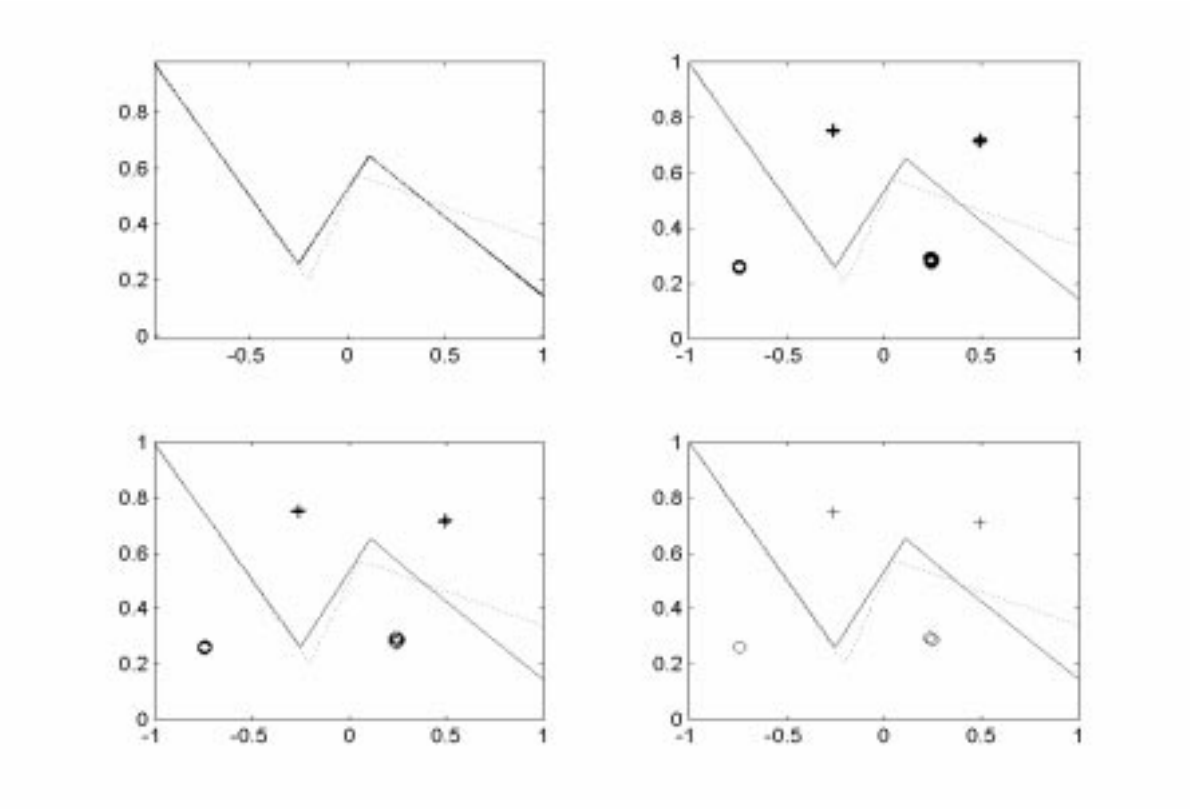


Fig.11. Local extreme for the Ripley's Problem: top left) several solutions computed with the LVQ1 algorithm (Note that they are almost identical); top right) the fussed codebook of 100 training sessions and the NN class border using this codebook; bottom left) the remaining codebook once we apply step 2.1 of local extreme and the class border for these prototypes; bottom right) The final codebook of local extreme and its class border. We also show the Bayes border (dotted line).

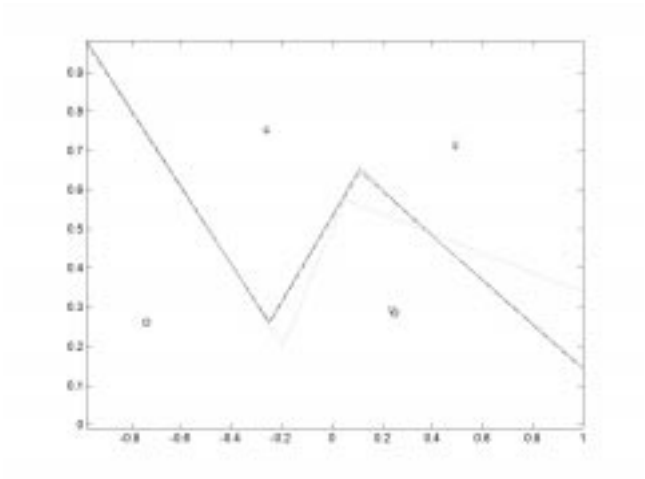


Fig.12. Local averaging vs. local extreme. We show Bayes border (dotted line), local averaging border (continuos line) and local extreme border (discontinuous line), the local averaging prototypes (diamonds and squares) and the local extreme prototypes (circles and crosses). Note that both algorithms achieve virtually the same solution.

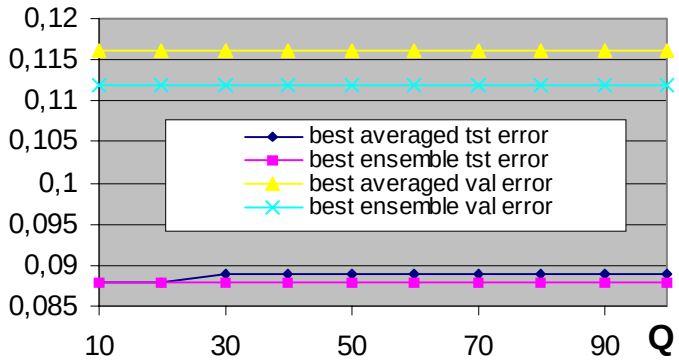


Fig. 13. Misclassification rate vs. the number of averaged codebooks in the Ripley's problem

4.3.3. 1D artificial Problem

Suppose two equally probable gaussian classes A and B ($N(m_A,\sigma)$ and $N(m_B,\sigma)$)with the same variance σ (2.5) and centres m_A=2 and m_B=7. For this problem, the optimal set of prototypes of a nearest neighbour classifier is formed by two prototypes w_A and w_B which induce a frontier point F=(w_A+w_B)/2 equal to the Bayes point ((m_A+m_B)/2=4.5). Assume now that we compute a series of codebooks using LVQ1 in different training sessions in which we vary the initial conditions and the order of training samples in memory. As we show in (Bermejo, 2000a), the attractor points of LVQ1 depend on the order of samples in memory so the computed predictors vary from a training session to another. This behaviour can be modelled assuming that theses predictors belong to gaussian random variables. Consequently, we will denote $\hat{w}_A$ and $\hat{w}_B$ as the predictors computed in the training sessions which are random samples belonging to $N(\overline{w}_A,\sigma_1)$ and $N(\overline{w}_B,\sigma_2)$. The expectations of these predictors will be 1 and 8 since these are the empirically observed attractors of LVQ1 for this problem (see (Bermejo, 2000a) §5.3). We have generated an ensemble of 120 NN classifiers where their two prototypes are random samples of the above distributions. Then we have applied local averaging and local extremes to compute a stabilised frontier point F. We have repeated these computations for different values of σ_1 and σ_2. Finally, the whole process has been reproduced for 100 different random sets of the prototypes. Figure 14 shows the mean and variance (respect to the 100 different random sets) of the relative errors in the P(Err\$C_2$) and F. As one can see, local averaging clearly outperforms local extreme since its mean relative error and variance is smaller. However local extremes behaves not so bad when $\sigma_1 = \sigma_2$. Figure 15 shows mean and variance of these relative errors for this particular case. The accuracy of local extremes is extremely degraded as the sigma augments. Only for small values of sigma it remains

competitive. Suppose now that another sub-optimal solution arises in the training sessions. This sub-optimal solution is centered at –1.85 and 10.5 (F=4.325). We model this new solution as before where we only consider the case in which the four distributions have the same variance σ. As figure 16 shows the interference notably affects local averaging while local extreme remains unaffected. However, as before, The instability of local extreme augments as sigma is increased.

5. Discussion

Local averaging achieves a stabilised NN classifier from an ensemble by means of: 1) the generation of an ensemble of Q perturbed codebooks of size K using a LVQ algorithm with different initial conditions and ordering of training data and 2) the local application of the averaging technique used in regression. The resulting classifier has a single codebook that is formed with the K centroids that are the outcome of applying the batch K-means algorithm over the totality of QxK codevectors. This method can improve in some cases the classification accuracy since it reduces the variance of codevectors as the result of averaging an ensemble of particular 'bootstrap' replicates. Hence, local averaging pretend to stabilise implicitly the class borders through an explicit stabilisation of the prototypes.

Local extreme gets those prototypes from the fussed codebook (that is generated by adding the totality of prototypes computed in the ensemble) that form class borders. Local extreme tries to stabilise class border in an explicit way using the extreme prototypes of each class from the ensemble. As we have shown in the experimental section, this method has a greater instability than local averaging, since its behaviour can notably vary between different ensembles in some cases. However if some very different solutions co-exist in the ensemble, local extreme can be useful.

6. Conclusions

We have introduced two simple methods that generate a single classifier from an ensemble of Nearest Neighbour (NN) classifiers. They are based on getting an ensemble of perturbed NN classifiers using a Kohonen's LVQ algorithm that is initialised with different ordering of training data and different initial conditions.

The first method (*local averaging*) stabilises class borders implicitly through an averaging of the prototypes of the ensemble. Since the distribution of the codebooks of the NN ensemble is arranged in clusters, a simple method of stabilisation is proposed based on computing an average solution in each natural group using the batch K-means. As we have shown, this method can be

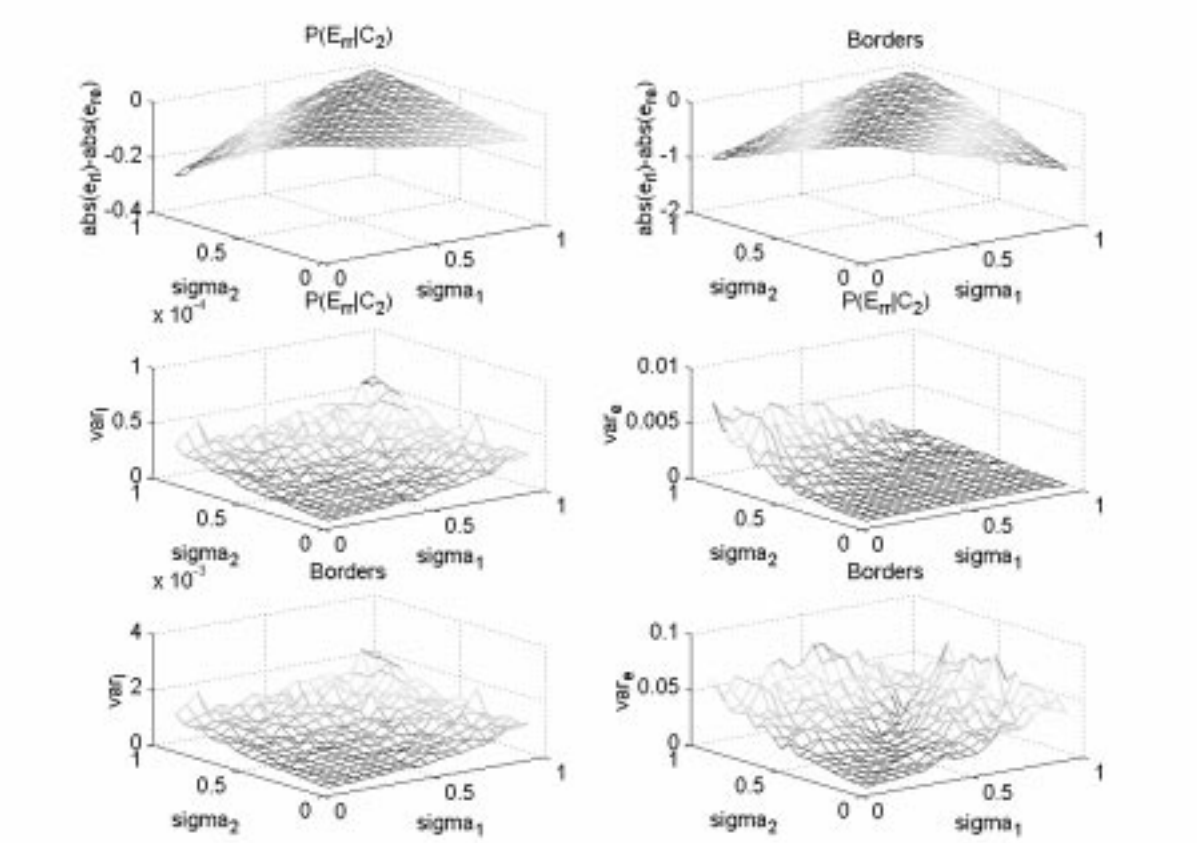
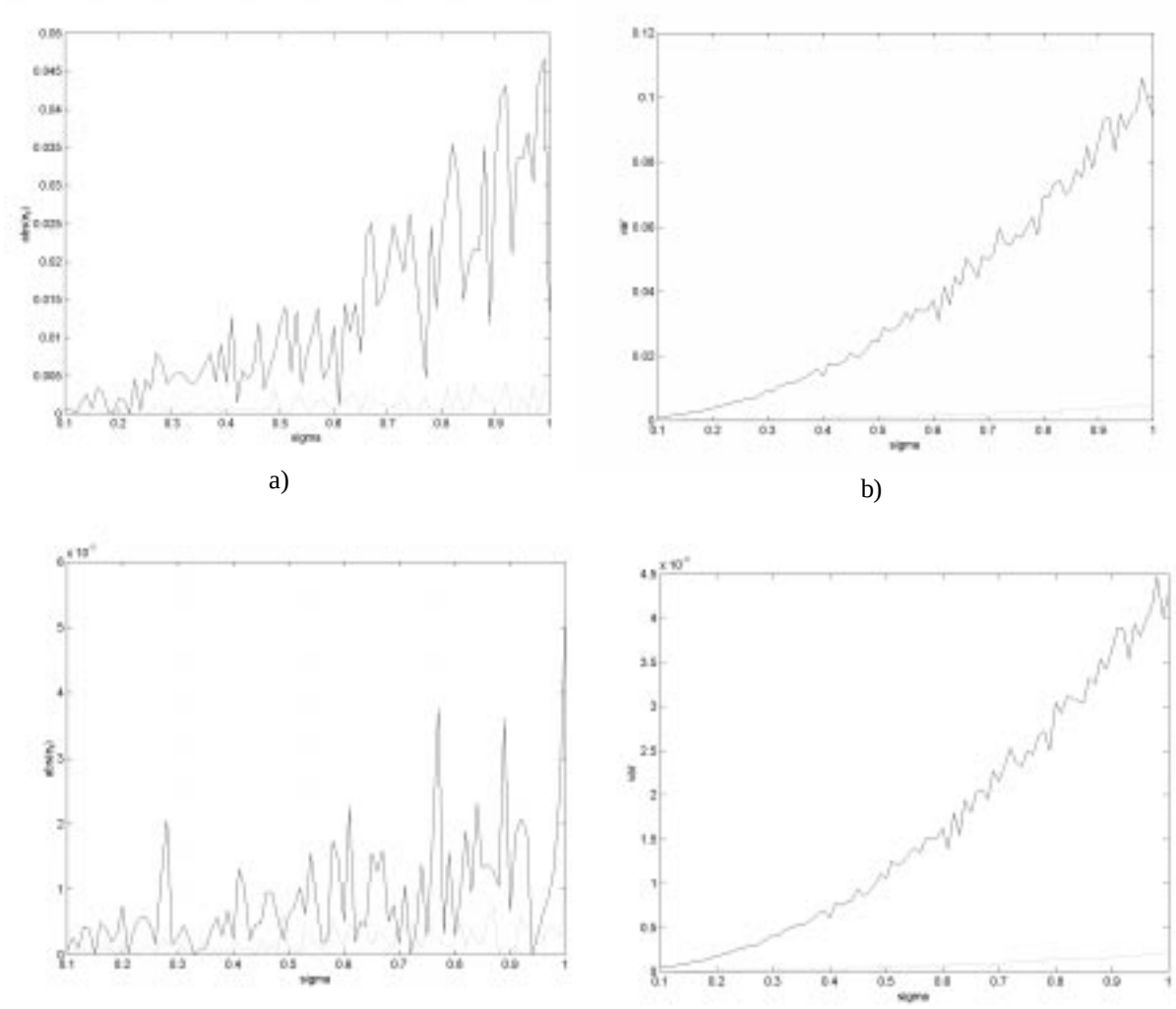


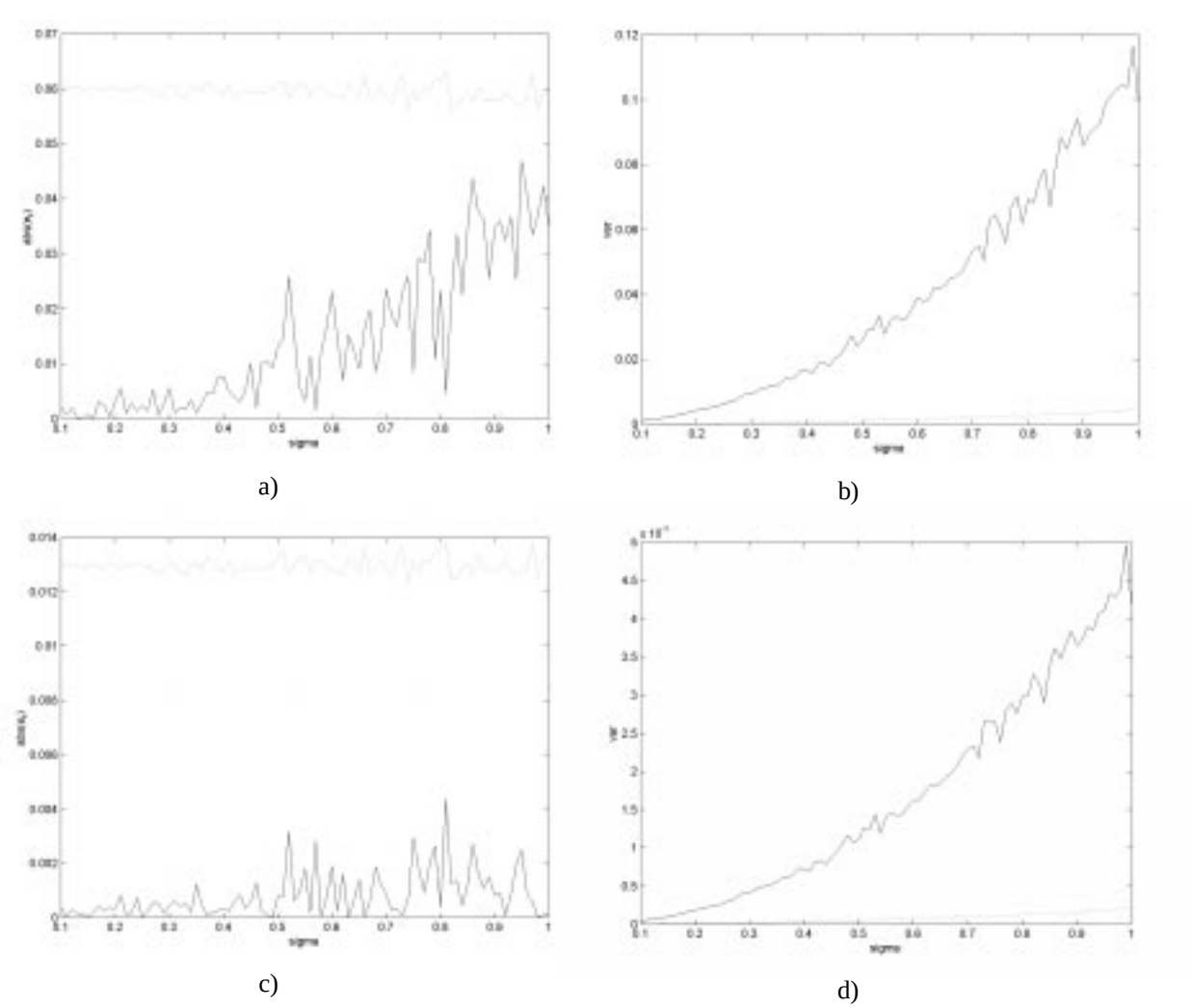
Fig. 14. Local averaging vs. local extreme in the 1-D artificial problem. We denote e_{rl} as the mean of the relative error of the local averaging method E_{rl}, e_{re} as the mean of the relative error of the local extreme method E_{re}, var_l as the variance of E_{rl} and var_e as the variance of E_{re}. These errors are calculated for the probability of error of class 2 $P(Err\backslash C_2)$ and for the frontier point as 1-(achieved/optimal). We compute this statistics generating 100 different sets with 120 random points for each prototype. The distribution of each prototype is gaussian with centers at 1 and 8 and variance $sigma_1$ and $sigma_2$ respectively. This distributions emulate the dispersion of the computed protoypes for different training sessions in which e.g. we vary the order of the training point in memory. Note that local extreme is only near to the accuracy of local averaging when the distribution of both prototypes has the same variance. Besides, the variance of local extreme respect to the use of different random samples of the prototypes is much greater than local averaging's. (Note: abs() indicate absolute value.)



a) b)

c) d)

Fig. 15. Local averaging (discontinuous line) vs. local extreme (continuos line) in the 1-D artificial problem. We compute the mean value and the variance of the relative error of $P(Err\backslash C_2)$ (a and b) and the frontier point (c and d). This statistics are computed generating 1000 different sets with 120 random points for each prototype. The distribution of each prototype is gaussian with centers at 1 and 8, and the same variance sigma for both distributions. Note that the local extreme method has a unstable behavior since its variance augments as sigma. Besides, local averaging is clearly superior since its relative error is smaller. However, *local extreme* for small values of sigma remains competitive.



a) b)

c) d)

Fig. 16. Local averaging (discontinuous line) vs. local extreme (continuos line) in the 1-D artificial problem with interference in the solutions. We compute the mean value and the variance of the relative error of $P(Err\backslash C_2)$ (a and b) and the frontier point (c and d). This statistics are computed generating 1000 different sets with 100 random points for each prototype with gaussian distribution centered at 1 and 8, and the variance sigma and 20 points that denote an interference (a sub-optimal solution) centered at –1.85 and 10.5. Here the interference notably affects local averaging while local extreme remains unaffected.

viewed as either a technique to reduce the variance of codevectors or as the result of averaging a series of particular bootstrap replicates.

The second method (*local extreme*) is based on the idea of generating a single codebook from the extreme prototypes of the whole set of codebooks. Local extreme tries to stabilise class borders explicitly using these extreme prototypes.

While local averaging is less sensitive to the particular ensemble of codebooks used for computing the combined codebook, local extreme allows combining codebooks of different sizes in a natural way and also can lead to better solutions in problems where several solutions co-exist.

Simulations in seven classification problems demonstrate the potential of the proposed local stabilising methods.

References

Avnimelech, R. & Intrator, N. (1999). Boosted Mixture of Experts: An Ensemble Learning Scheme. Neural Computation, 11.

Bermejo, S. (2000a). Finite-Sample Convergence Properties of the LVQ1 algorithm and the BLVQ1 algorithm, in Bermejo, S. Learning with Nearest Neighbour Classifiers, Ph.D. Thesis, Barcelona: Universitat Politècnica de Catalunya.

Bermejo, S. (2000b). A Dynamic LVQ algorithm for improving the generalization performance of Nearest Neighbour Classifiers, in Bermejo, S. Learning with Nearest Neighbour Classifiers, Ph.D. Thesis, Barcelona: Universitat Politècnica de Catalunya.

Bottou, L. (1998). Online Learning and Stochastic Approximations, in David Saal (Ed.) Online Learning and Neural Networks, Cambridge: Cambridge University Press.

Breiman, L. (1998). Half- &-Half Bagging and Hard Boundary Points. Technical Report No.534, Berkley: University of California, Department of Statistics.

Breiman, L. (1994). Bagging Predictors. Technical Report No.421, Berkley: University of California, Department of Statistics.

Efron, B. & Tibshirani, R.J. (1994). An Introduction to the Bootstrap. Chapman & Hall.

Garris, M. et al. (1997). NIST Form-Based Handprint Recognition System (Release 2.0), National Institute of Standards and Technology.

Geman, S. Bienenstock, E. & Doursat R. (1992). Neural Networks and the bias/variance dilemma. Neural Computation, 4.

Gersho, A. & Gray, R.M. (1992). Vector Quantization and Signal Compression. Kluwer Academic Publishers.

Hansen, L.K. & Salamon, P. (1990). Neural Network Ensembles. IEEE Trans. on PAMI, 12.

Jacobs, R.A. et al. (1991). Adaptive Mixtures of Local Experts. Neural Computation, 3, 79-87.

Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., & Torkkola, K., LVQ_PAK. The Learning Vector Quantization Program Package. Version 3.1, Helsinki: Helsinki University of Technology, Laboratory of Computer and Information Science.

Kohonen, T. (1996). Self-organizing Maps, 2nd Edition, Berlin: Springer-Verlag.

LaVigna, A. (1990). Nonparametric classification using LVQ. Ph. D. Dissertation. University of Maryland.

Murphy, P. M., & Aha, D. W. (1994). UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. Department of Information and Computer Science, University of California.

Natftaly, U., Intrator, N. & Horn, D. (1997). Optimal Ensemble Averaging of Neural Networks. ??

Perrone, M.P. (1993). Averaging Techniques for Neural Networks, in Arbib, M.A. (Ed.), The Handbook of Brain Theory and Neural Networks, Boston, MA: MIT Press.

Partridge, D. & Yates, W.B. (1996). Engineering Multiversion Neural-Net Systems. Neural Computation, 8.

Ripley, D. (1994). Neural Networks and Methods for Classification, Journal of the Royal Statistical Society, Series B, 56, p. 409-456

Schapire, R.E. (1999). A Brief Introduction to Boosting, Proceedings of the 16th International Joint Conference on Artificial Intelligence.

Wolpert, D. (1992). Stacked Generalization, Neural Networks, 5.