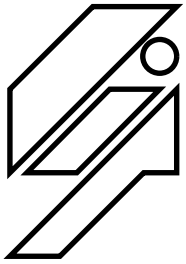


Laboratoire de l'Informatique du Parallélisme

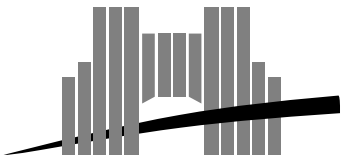
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

Comparative study of three connectionist models on a classification problem

R.Baron, M.B.Gordon,
H.Paugam-Moisy, J-M.Torres
Moreno

April 12, 1996

Research Report N^{O} 96-07



Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) 72.72.80.00 Télécopieur : (+33) 72.72.80.80
Adresse électronique : lip@lip.ens−lyon.fr

Comparative study of three connectionist models on a classification problem

R.Baron, M.B.Gordon, H.Paugam-Moisy, J-M.Torres Moreno

April 12, 1996

Abstract

In this paper, we compare three neural learning models, on the sonar target classification problem of Gorman and Sejnowski, in two ways. First, we compare the performance on training and testing sets, in the usual sense. Second, we investigate in the database, in order to compare the examples which are hard to be learned and the patterns which are ill-classified, in generalization. The neural networks involved in these comparisons are a classical multilayered network, a wavelet network, and a evolutive architecture named "Monoplane". In spite of several differences in the characteristics of their learning algorithms, these neural networks perform similarly on the classification problem, and the patterns which are hard to classify are nearly the same, whatever the model be. Several distance measures are then computed on the database, but we show that none of them allows to explain the misclassification phenomenon. We conclude that hard to classify patterns depend on the database more on than the learning model.

Keywords: classification, learning, neural networks, Monoplane, wavelet networks

Résumé

Dans cet article, trois modèles connexionistes sont comparés, sur le problème de classification des signaux sonar de Gorman et Sejnowski. Tout d'abord, les performances en apprentissage et en généralisation sont comparées. Puis, la base d'exemple est analysée, afin de caractériser les exemples mal appris et mal reconnus. Les trois modèles mis en œuvre sont, un réseau multi-couches classique, un réseau d'ondelettes, et une architecture incrémentale, appelée "Monoplan". Malgré les différences des algorithmes d'apprentissage utilisés, ces trois modèles se comportent de manière similaire pour la tâche de classification, et les exemples mal classés sont a peu près les mêmes, quel que soit le modèle. Des mesures de distances sont alors utilisées, mais aucune d'elles ne permet d'expliquer complètement les cas de mauvaise classification. Nous concluons que les exemples difficiles à classer dépendent de la base d'exemple, plus que du modèle mis en œuvre.

Mots-clés: classification, apprentissage, réseaux de neurones, Monoplan, réseau d'ondelettes

Comparative study of three connectionist models on a classification problem

R.Baron[‡], M.B.Gordon[†], H.Paugam-Moisy[‡], J-M.Torres Moreno[†]
[†] CEA/Département de Recherche Fondamentale sur la Matière Condensée
17 rue des Martyrs, 38054 Grenoble Cedex 9, France
gordon@drfmc.ceng.cea.fr
[‡] LIP - URA 1398 du CNRS - Ecole normale supérieure de Lyon
46 allée d'Italie, 69364 Lyon Cedex 07, France
rbaron@lip.ens-lyon.fr

April 12, 1996

1 Introduction

In real-world applications, for industrial, financial, or medical applications with large amounts of data, several questions arise to developers, such as the choice of the right method and the best model. From a theoretical point of view, multilayer neural networks and back-propagation learning, for instance, have been related to data analysis [2, 14], and to Bayesian probabilities [11]. However such results are not very helpful for practical applications. In literature, many authors compare their approach to other models, in terms of success on some benchmark or school-problem. Generally the differences are very tiny (in favor of the new model, obviously). Even between several connectionist models, applied to a same problem, the difference of performance is not always very significant. As a matter of fact, there does not exist a best network for a given application. The percentages of success cannot be strictly defined for a neural model, neither in learning phase nor in generalization. Only a set of good networks can be defined, with suitable architectures and ranges of good parameters [8]. Therefore, performance should better be given with an error bar, as physicists do.

Comparisons of models on benchmarks have been recently performed by various scientific communities [12, 3]. One of their interest is to emphasize the salient features of various methods, such as learning fastness, robustness, explanatory power, etc. However, limitations of performance are not always intrinsic to the models. They may rather be intrinsic to the databases. In this article, we compare three neural models on a sonar target classification problem, in two ways. First we compare the performance on training and testing sets, in the usual sense. The neural networks involved in these comparisons are a classical multilayer network, a wavelet network, and an evolutive architecture named "Monoplane". Second we investigate in the databases, in order to compare the examples which are hard to be learned and the patterns which are classified in the wrong class. Several distance measures are computed on the database, but we show that none of them allows to explain the misclassification phenomenon.

2 Wavelet neural network

2.1 Principle and architecture

Recently, the theory of wavelet decomposition has been related to the neural network area [1]. The wavelet theory allows to decompose a function f, using a family of functions obtained by dilating and translating a single wavelet function. In the discrete case, it can be shown that f is approximated as follows:

$$f(x) \approx \sum_{i=1}^{N} w_i \cdot h(d_i(x - t_i)) \tag{1}$$

where d_i and t_i stand respectively for dilation and translation factors, and h is a wavelet function, i.e. the Fourier transform of which verifies some conditions.

In this study, a neural network is applied from $\mathbb{R}^n$ to $\mathbb{R}$. A multi-dimensional wavelet $\Psi : \mathbb{R}^n \mapsto \mathbb{R}$ is built by setting $\Psi(x) = (x^T \cdot x - n).exp(-\frac{1}{2}.x^T \cdot x)$ where n is the dimension of the input space, as in [15]. A continuous wavelet decomposition formula in the multidimensional case can be derived from the scalar case. A discrete approximation for a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is

$$s(x) = \sum_{i=1}^{N} w^{(i)} \cdot \Psi(D^{(i)}(x - t^{(i)})) + \theta \tag{2}$$

This expression may be written as the output of a *wavelet neural network* with one hidden layer, as shown in figure 1. The network output is computed from formula (2) where

- $D^{(i)}$ is a dilation matrix, which is a diagonal matrix (built with a dilation vector $\{d_j^{(i)}\}_{1 \leq j \leq n}$ on its diagonal, and 0 elsewhere);
- $t^{(i)}$ is a translation vector;
- θ is a parameter introduced to help dealing with non-zero mean functions (the average value of Ψ is 0, thus the average value of s is θ).

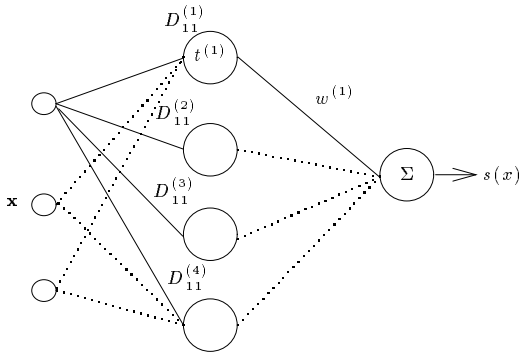


Figure 1: Architecture of a wavelet network.

Since $t^{(i)}$ and θ appear as biases, they can be considered as weights of the model, as well as $w^{(i)}$ and $d_j^{(i)}$. The output cell computes a weighted sum of hidden cells activities, and may apply a sigmoidal transfer function if the network is to be used for a classification task.

2.2 Learning algorithm

Each element of a database $\mathcal{S}$ is an input/output pair $(x_k, f(x_k))$, where f is the function to be approximated or the class to be assigned. For every weight, the learning rule is a gradient descent algorithm, like in back-propagation learning. The cost function to be minimized at each step is $c(x_k) = \frac{1}{2}(s(x_k) - f(x_k))^2$. The database $\mathcal{S}$ is split into a training set $\mathcal{S}_t$ and a generalization set $\mathcal{S}_g$. After each presentation of a pair $(x_k, f(x_k))$ from $\mathcal{S}_t$, each weight X of the network (X stands for $w^{(i)}, d_j^{(i)}, t^{(i)}$ *or* θ) is modified according to $X_{k+1} = X_k - \gamma . grad_X c(x_k)$. The performance of a wavelet neural network for approximating a function is given by the total cost (in generalization phase), which is the mean of $c(x_k)$ for x_k in $\mathcal{S}_g$. If the wavelet network is applied to a classification task, the performance is the amount of success in correctly assigning its class to each pattern of $\mathcal{S}_t$ (in learning phase) or $\mathcal{S}_g$ (in generalization).

3 Monoplane, an evolutive neural network

The Monoplane algorithm has been described in detail in [13]. It generates a neural network with one hidden layer of binary units, and a binary output unit.

3.1 Learning algorithm

First, a binary perceptron is trained from the learning set. If the training performance $P_t = 100\%$, the learning set is linearly separable and the algorithm stops. If $P_t < 100\%$, we add a hidden unit h, and train it to give an output $\sigma_h = 1$ for all those patterns that were correctly classified and $\sigma_h = -1$ for patterns misclassified by the previous unit. Thus, each new hidden unit helps to correct part of the errors produced by the previous ones. Once the training of a hidden unit is over, its weights are freezed, and not modified any more. The hidden layer grows until one unit learns all its σ_h correctly. It has been shown by Martinez and Esteve [6] that this procedure converges, and that the output is the parity of the internal representations associated to the training set. This first stage of Monoplane is identical to the first layer construction of their Offset algorithm. It differs in that the latter needs a second hidden layer.

Second, the output unit is connected to the units of the hidden layer. This unit is trained to learn the desired outputs, on the training set. If the output unit does not find a solution without errors, we go on adding hidden units to correct the output errors, i.e. we trained to learn $\sigma_h = 1$ for all those patterns that were correctly classified and $\sigma_h = -1$ for patterns misclassified by the output unit. This happens if the internal representations are not linearly separable. Hence, new hidden units are included one after the other, until one unit learns its training set without errors. Then we try again to learn the desired outputs.

3.2 Main features of the model

Monoplane reduces the learning problem to that of training single perceptrons, as many as hidden units are needed, plus the output. Its overall performance is strongly dependent on the perceptron learning algorithm. We use Minimerror, which is based on the minimization of a cost function that may be interpreted as a noisy measure of the training error. It has been theoretically shown, and experimentally confirmed, that this algorithm has optimal performances both in training and in generalization [4, 10].

4 Application and comparison of models

Three connectionist learning models have been tested on a sonar target classification: Monoplane, back-propagation, wavelet learning. The database $\mathcal{S}$ gathers a file "sonar.mines" which contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles (and under various conditions), and a file "sonar.rocks" of 97 patterns obtained from rocks under similar conditions. This base has been used in an earlier work by Gorman and Sejnowski [5], and then broadcasted on Internet. $\mathcal{S}$ can be decomposed for two series of experiments. First an "aspect-angle independent" series (further denoted by AAI), in which the whole data set is used without controlling for aspect angle, and an "aspect-angle dependent" series (further denoted by AAD) in which the training and testing sets are carefully controlled to ensure that each set contains cases from each aspect angle in appropriate proportions. In our experiments, each pattern is a set of 60 real inputs, and one binary desired output (-1 for a mine and +1 for a rock).

4.1 Results on AAI experiments

For AAI series, 13 packets of 16 randomly chosen patterns have been drawn. Hence, 13 series of experiments can be performed, each with a different packet of 16 patterns in $\mathcal{S}_g$, and the 192 remaining patterns as examples, in $\mathcal{S}_t$. Means and error bars presented on figure 2 are obtained by averaging the performance of the 13 series of training and generalization sets. Results are presented for various numbers of hidden units. Monoplane algorithm stops for different numbers of hidden units, depending on the series of example sets.

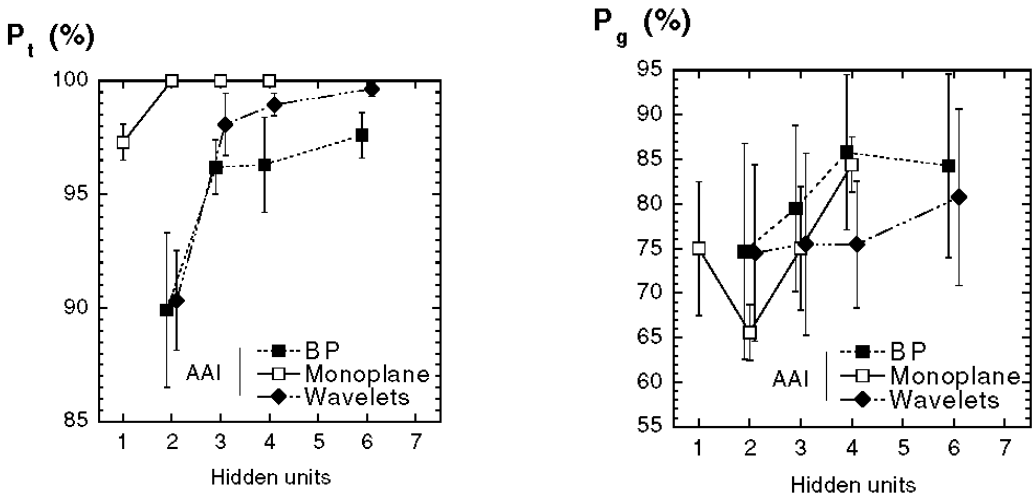


Figure 2: Experimental results, on average, for the AAI series

Learning performance is maximum for Monoplane (inherent in the model), and slightly better for wavelet networks than for multilayer networks. Nevertheless the curves of generalization are very similar and confuse, with a slightly better performance for multilayer networks.

4.2 Results on AAD experiments.

For AAD series, both the learning set $\mathcal{S}_t$ and the testing set $\mathcal{S}_g$ are composed of 104 patterns which are supposed to be uniformly distributed.

Multilayer networks are trained during 500 epochs[1], the learning set $\mathcal{S}_t$ being mixed before each epoch. Generally, the networks became stable between 200 and 300 epochs, except for too small numbers of hidden units. Wavelet neural networks are trained towards 30 000 iterations, with random choice of an example in $\mathcal{S}_t$ at each iteration. Since $\mathcal{S}_t$ contains 104 examples, the number of iterations is nearly the same for the convergence of both models. Monoplane learns towards a perfect success, according to the definition of the model. The only exception, on this diagram, is the case of a network without hidden layer.

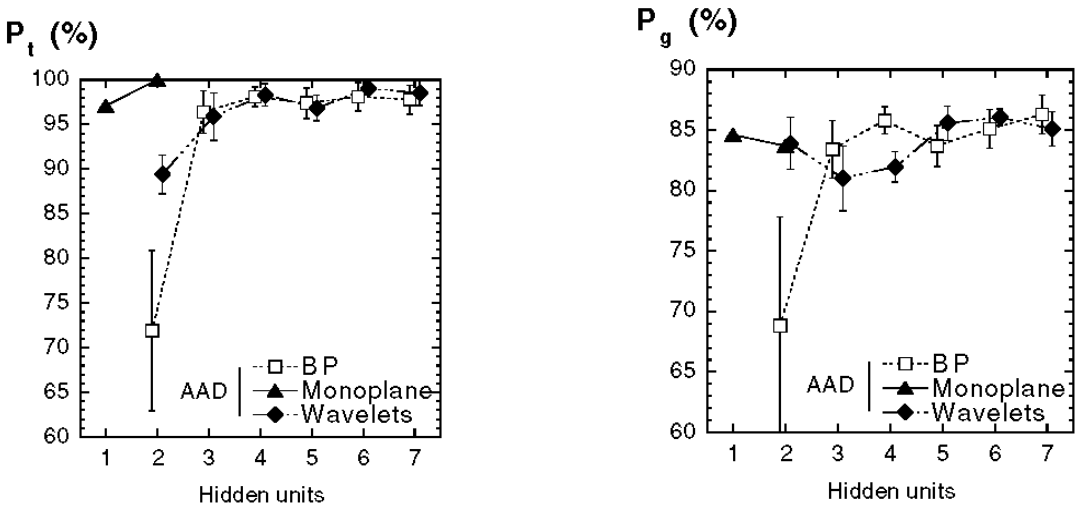


Figure 3: Experimental results, for the AAD series.

For each model, except for Monoplane which is deterministic, several initial sets of weights, or several learning rate values have been tested, for different numbers of hidden units. The two graphics of figure 3 show average success in learning and generalization phases respectively, for the three models. Error bars represent the dispersion.

Once again, performance is roughly independent of the model. The curves obtained for back-propagation and for wavelet learning are confused. Monoplane performs better in the learning phase, with a very low number of hidden units, but its performance in generalization is not so high than the best values of the two other models. A better tuning of Monoplane parameters allows to find a network, without hidden units, which performs 100% on the learning set, but slightly worst on $\mathcal{S}_g$ than the graphic shows. This result means that the 104 patterns of $\mathcal{S}_t$ are linearly separable. The most important result is that the set of 208 patterns is also linearly separable. This property (certainly unknown from Gorman and Sejnowski) has been discovered in running the Monoplane learning algorithm. A learning experiment has been performed on the whole data base $\mathcal{S}_t \cup \mathcal{S}_g$ and a network with no hidden units which performs 100% has been found. The hyperplane H_0 defined by this network will be used as a reference in a further section.

[1] an epoch is a complete presentation of the whole training set $\mathcal{S}_t$.

4.3 Comparison of the models

In spite of their common behavior on experiments, the three learning models differ in several characteristics. Whereas multilayer networks and Monoplane learning algorithms are based on separating the space by hyperplanes and then applying non-linearities, wavelet networks operate local clustering (like RBF networks do) and then apply non-linearities. On the other hand, whereas multilayer and wavelet networks learn from a global cost-minimization of the whole network (every weight at each iteration), Monoplane learns units one by one and minimizes the cost function for one unit after the other, in a constructive way. These differences allow to guess that the three models would not capture the same features from the database. Thus our next experiment has been to compare the examples which are difficult to be learned and the hard to classify patterns, in generalization. Next sections present the results of these comparisons in the case of the AAD experiments.

5 Patterns which are difficult to well classify

5.1 Ill-learned examples

Since the learning success is always close to 100% (cf. figure 3), there are only few ill-learned examples, except for multilayer or wavelet networks without enough hidden units. For Monoplane, in the case of no hidden units, presented on figure 3, three examples are not well learned: number 53, number 57, and number 66. For the two other models, ill-learned examples have been associated to the percentage of their occurrences in all the experiments performed with different initial weights, or different numbers of units, or different learning rate values. The most frequent ill-learned example is number 1, for both multilayer (88%) and wavelet (54%) networks. Number 57 is their second more frequent ill-learned example: 25% for multilayer and 37.5% for wavelet networks; number 66 is also ill-learned: 21% for multilayer and 17% for wavelet networks.

In conclusion, among the short lists of ill-learned examples for the three models, two numbers appear frequently, whatever the model be. In the peculiar case of Monoplane, this question is no longer significant, since all the examples have been proved to be learnable, even without hidden units, when the tuning of the learning parameters is sufficiently sharp.

5.2 Ill-classified patterns, in generalization

Monoplane gives a unique list of ill-classified patterns, for each number of hidden layers, before reaching perfect learning. In generalization, the list for one hidden layer (2 hidden units) is the same as for no hidden layer, plus one pattern. For multilayer and wavelet networks, percentages have been calculated for the occurrence of ill-classified patterns, as explained for ill-learned examples. Once again, we established that the most frequently ill-classified patterns are nearly the same, whatever the learning model be. Table 1 presents the patterns which are the most difficult to be well classified, with an indication of their presence (for Monoplane) or their frequence (for multilayer and wavelet networks).

In this table, 12 patterns are present among the 17 patterns of the lists of ill-classified patterns for Monoplane. The table also contains all the patterns the frequence of which is greater than 60%, both for multilayer and for wavelet networks. The results show a strong correlation in ill-classified patterns, and a strong consensus, nearly independent of the learning model. We must conclude that: hard to classify patterns depend on the database more than on the learning model.

Pattern number	Monoplane	Multilayer network	Wavelet network
1	yes	100 %	75 %
2	no	63 %	79 %
5	yes	71 %	79 %
16	yes	17 %	25 %
29	yes	71 %	79 %
30	no	33 %	71 %
32	yes	67 %	46 %
35	yes	75 %	17 %
37	no	92 %	63 %
38	no	100 %	71 %
66	yes	63 %	46 %
68	yes	67 %	25 %
81	yes	42 %	75 %
82	no	71 %	79 %
93	yes	33 %	4 %
97	yes	67 %	13 %
98	yes	25 %	33 %

Table 1: Most frequently ill-classified patterns, in generalization, for AAD series.

5.3 Distances between examples

Next issue is to investigate in the database itself in terms of distances, in order to find an explanation to misclassification. For each pattern, the minimum and the maximum distances to all the other patterns in $\mathcal{S}_g$ have been computed. Afterwards, patterns have been sorted, first by increasing minimum distance, second by increasing maximum distance. Patterns the number of which is in table 1 have been located in both lists. For the minimum distance, more than half the set of ill-classified patterns is concentrated in the last third part of the list. All these patterns, except pattern number 2, are also in the first half part of the maximum distance list. That means they probably are far from the other patterns of their class, and close by patterns of the opposite class. It is a good reason for being ill-classified. Nevertheless, patterns number 16, number 35, number 97, and number 98 have a low minimum distance and a high maximum distance, which seems to be paradoxical. It is interesting to notice that those critical patterns are more weakly ill-classified by wavelet networks than others, probably due to the local clustering learning.

6 Removing conflicting data

One problem with real-world data bases lies in conflicting data. Input/output pairs are said to be *conflicting* pairs when similar inputs are mapped into dissimilar outputs. Following [7], a measure is proposed. The purpose is to measure the degree of conflict between two input/output pairs. Assuming these pairs are denoted by $(\vec{x}_k, \vec{y}_k)$ and appropriate distance metrics $d_{\vec{x}}$ and $d_{\vec{y}}$ are given respectively in input and output vector spaces, the function

$$s(i,j) = \frac{d_{\vec{y}}(\vec{y}_i, \vec{y}_j)}{d_{\vec{x}}(\vec{x}_i, \vec{x}_j)}$$

is computed. Euclidean distance metric will here be used for $d_{\vec{x}}$ and $d_{\vec{y}}$. Then, s can be interpreted as a conflict measure between pairs i and j: the higher the value of s, the higher the degree of conflict between pairs $(\vec{x}_i, \vec{y}_i)$ and $(\vec{x}_j, \vec{y}_j)$. In order to make the learning phase easier, this measure can be used in a conflicting data removing procedure: given a threshold Θ, when $s(i,j) > \Theta$, both examples i and j are removed from the learning base. This may prevent a neural network from learning conflicting data, which violates the definition of a function (one input mapped to one and only one output).

6.1 Conflict measure and misclassified patterns

The conflict measure has first been used on the AAD generalization base $\mathcal{S}_g$. $s(i,j)$ have been computed for all the input/output pairs of this base. It was assumed that ill classified cases could be resulting from conflicting input/output pairs. Then, the removing procedure has been applied on the generalization set, in order to detect conflicting data: cases i and j for which $s(i,j) > threshold$ have been removed.

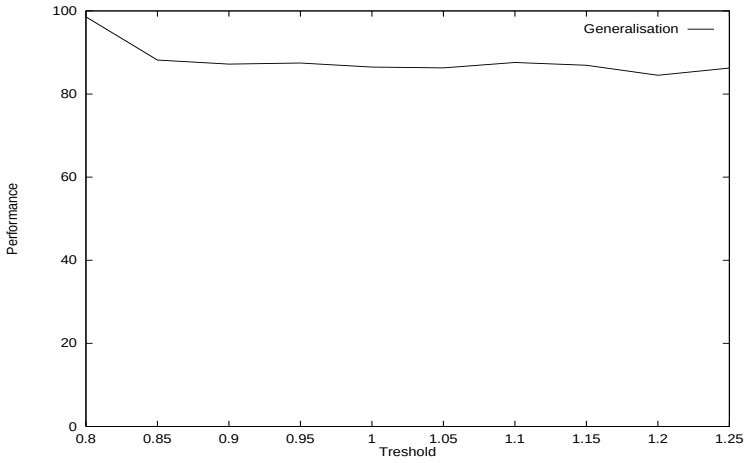


Figure 4: Partial generalization performance

Figure 4 plots the generalization performance on patterns which are not removed from the generalization set $\mathcal{S}_g$, as the threshold varies. This rate is roughly uniform, whatever the threshold be. The first rate, at a threshold 0.8, is higher, because of the small number of remaining patterns (only 7 cases). If assuming that removed examples are the hard to classify patterns, then, as the threshold increases, only very conflicting patterns are removed, and remaining examples should be easier to classify. Hence, generalization performance should be increasing with the threshold. This is not the case. We must conclude that for the sonar target classification, in generalization, the conflict measure s is not relevant for the ill-classification problem. Removed examples according to this measure, are not actually ill-classified patterns.

6.2 Conflict measure and learning

In order to find a better explanation, new experiments have been performed. The conflict measure s was first proposed to ease the learning phase. The measure is used to remove conflicting pairs from the learning set. Influence of both the threshold and the number of cases in this base has been studied. Therefore, all the example bases have to be redefined. $\mathcal{S}_t \cup \mathcal{S}_g$ is used to build three nested training sets containing 52, 104 and 156 examples, and a common generalization set with

52 patterns. The following procedure is then applied to each of the training sets: given a threshold value Θ, input/output pairs i and j for which $s(i,j) > \Theta$ are removed from the learning base. The learning algorithm is applied with the remaining examples. We call "original training set" the set before the removing procedure is applied, and "learning base", the group of remaining examples after the deletion (that is, examples which are effectively learned). Results are displayed below. On each figure, three curves are given, according to the size of the original training set (52, 104 and 156 cases).

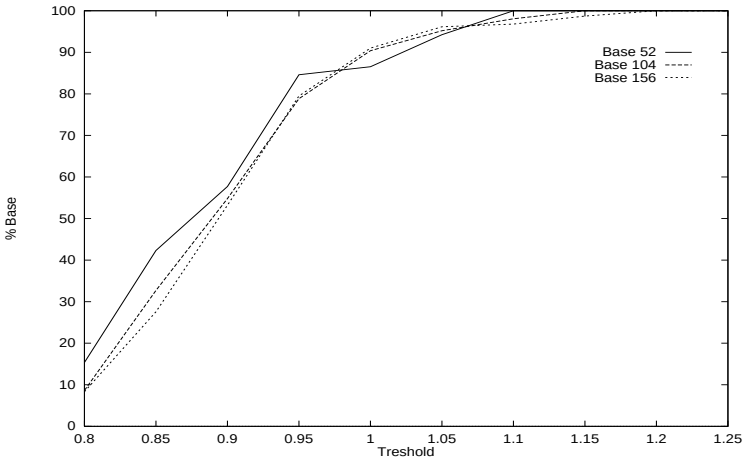


Figure 5: Percentage of remaining examples

Figure 5 plots the amount of the original training set used in the learning procedure (i.e. the ratio #(learning base)/#(original training set)). The three curves have the same shape. But the threshold for which 100% of the set is kept for training, slightly increases with the size of the original training set. Thus, the degree of conflict in the training set remains nearly the same, whatever the size of the original training set. However, the number of conflicting pairs logically increases with the size of the original set. The three training bases can be considered as equally representative of the whole database including 208 examples.

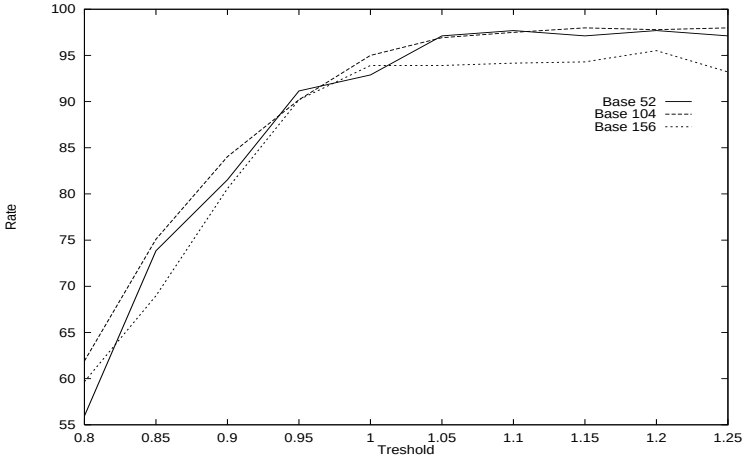


Figure 6: Classification on the original training set

Figure 6 shows the rate of well classified examples on each original training set. Depending on

the threshold value, some examples are included in the learning base, some are excluded since they are conflicting. As in the first figure, the three curves have the same shape, and are nearly identical. This is a consequence of the previous observation. The rate of remaining examples depends on the threshold value, more than on the original set size. And the rate of well learned examples is also mainly dependent on the threshold value. Thus, the rate of success on the original training set is nearly the same, whatever the original set size be.

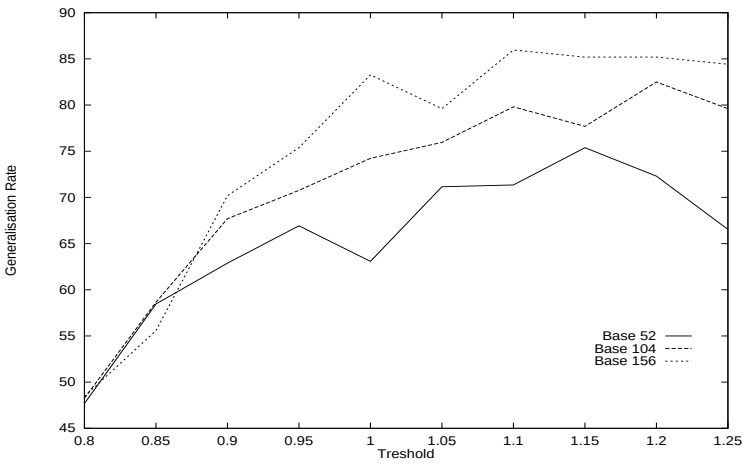


Figure 7: Generalization rate

The main difference between the three bases is given in figure 7, which plots the generalization rate. As it could be expected, the rate globally increases with the threshold, since a larger number of examples has been learned. But this rate also increases with the size of the original learning base. Learning patterns are not so numerous in order that the learning base is redundant. Therefore, the greater the learning base, the better the generalization performance, in the limit of the available database for this application.

In conclusion, the conflict measure s has not been proved to be a very relevant criterion. It is not sufficient for characterizing ill-learned nor ill-classified patterns.

7 Measures and decision border

Another distance criterion could be the *stability* defined with regards to the Monoplane model [13]. The *stability* γ of a pattern i measures the distance of pattern i to the separating hyperplane, with positive sign if it is well classified, negative otherwise. A small value of the stability implies that an example is close by the separating hyperplane (see fig. 8). We may assume that, the smaller the stability of example i, the harder the correct classification of i.

Let us compare the conflict measure and the stability. The conflict measure $s(i,j)$ is higher as two patterns i and j are more similar, although belonging to different classes.

The stability of all the examples has been computed, from the hyperplane separating the whole base, as discussed in section 4.2.Notice that, for this application, the stability is positive for every pattern, since H_0 realizes a linear separation of the whole database. For the wavelet network, half of the hard to be learned pattern have a high stability value, hence they should have been easy to learn. However, the most difficult examples, number 57 and 66 have actually a very small stability. This may explain why these patterns are hard to be learned for both wavelet and multi-layered networks.

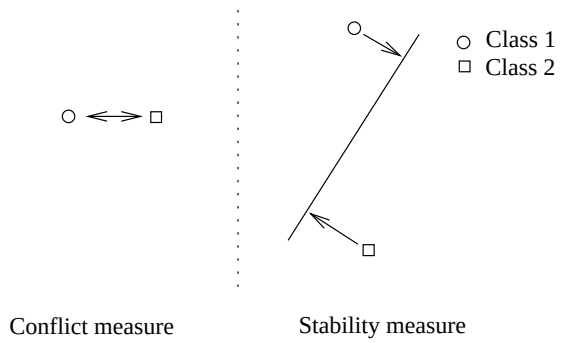


Figure 8: Comparison between the two distance criteria

Let consider now the generalization set and the table of ill-classified patterns for the three models (table 1, section 5.2). A proportion of 82% of these ill-classified patterns have a stability $\gamma < 0.1$. It has to be compared to the whole generalization set, for which the proportion of patterns with a stability $\gamma < 0.1$ is only 38%. The stability criterion of Monoplane is better than the conflict measure for explaining the patterns which are ill-classified, both by wavelet and multilayered networks.

The stability could be even more relevant in explaining multi-layered network misclassification, because the decision borders of this model are also hyperplanes. [Experiments are in progress in this direction.]

8 Conclusion

Three neural learning models have been compared on the sonar target classification problem of Gorman and Sejnowski [5]. In spite of several differences in the characteristics of their learning algorithms, these neural networks perform similarly on the classification problem. Experiments even prove that the examples which are hard to be learned and the patterns which are difficult to be well classified in generalization are nearly the same, whatever the model be. We have already encountered this phenomenon, in case of an industrial application of pollution prediction, and with different neural models (Kohonen topological maps and multilayer networks) [9]. We claim that hard to classify patterns depend on the database more than on the learning model.

Three measures of distance have been used to help dealing with learning failure: a classical euclidean distance, a conflict measure, and a stability measure. Neither the euclidean distance nor the conflict measure have been proved to be relevant to characterize patterns which are hard to learn, or hard to classify in generalization. The stability measure is more relevant than the previous criteria. This statement confirms the similar behavior which has been observed for the three models. A calculus of stability from hyperplanes computed by Monoplane in learning on the whole database, could be used as a reference for predicting hard to classify patterns. Nevertheless, the sonar application has proved to be a linearly separable classification problem, which is not the general case for real-world applications. A general criterion which would allow to distinguish hard to classify patterns is still a open problem.

Consequences of this work are both theoretical and practical. The theoretical aspect concerns the need of researching a definition of complexity for an application or for a database, a theory like the VC-dimension for the families of functions or networks, but with more realistic considerations. The practical consequences consist in pointing out the importance of the data collection phase for

the successful development of any application (industrial, medical, financial ...). Databases must be as large as possible, without inherent contradictions and without non-representative outliers. For classification problems, the different classes have to be equally represented in the database. Thus, if the database is carefully collected, the developer can guess that the performance is strongly dependent of the complexity of his application (which cannot be avoid) but nor of the database neither of the learning model.

References

[1] A. Benveniste and Q. Zhang. Wavelet networks. *IEEE Trans. on Neural Networks*, 3(6):889–898, November 1992.

[2] P. Gallinari, S. Thiria, and F. Fogelman-Soulié. Multilayer perceptrons and data analysis. In *IJCNN-San Diego*, volume I, pages 391–399, 1988.

[3] O. Gascuel and P. Gallinari. Méthodes symboliques-numériques de discrimination. Rapport final d'activité, Projet Inter-PRCs, 1994.

[4] M.B. Gordon and D. Grempel. Leraning with a temperature dependent algorithm. *Europhysics Letter*, 29:257–262, 1995.

[5] R.P. Gorman and T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89, 1988.

[6] D. Martinez and D. Esteve. The offset algorithm: Building and learning method for multilayer neural networks. *Europhysics Letter*, 18:95–100, 1992.

[7] L.R. Medsker. *Hybrid Neural Network and Expert Systems*. Kluwer Academic Publishers, 1994.

[8] H. Paugam-Moisy. Parallel neural computing based on network duplicating. In I. Pitas, editor, *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks*, chapter 10, pages 305–340. John Wiley, 1993.

[9] H. Paugam-Moisy. Quelques principes méthodologiques pour le bon usage des méthodes connexionnistes. In *Les applications industrielles de la reconnaissance des formes*, Séminaire du programme européen COMETT II, pages 23–36, INSA de Lyon, 1994.

[10] B. Raffin and M.B. Gordon. Learning and generalization with minimerror, a temperature-dependent learning algorithm. *Neural Computation*, 7(6):1206–1224, November 1995.

[11] M.D. Richard and R.P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[12] S.B. et al. Thrun. The monk's problems, a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991.

[13] J.-M. Torres-Moreno, P. Peretto, and M.B. Gordon. An evolutive architecture coupled with optimal perceptron learning for classification. In M. Verleysen, editor, *Proceedings of ESANN'95*, pages 365–370, Brussels, 1995. D facto.

[14] A.R. Webb and D. Lowe. The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks*, 3(4):367–376, 1990.

[15] Q. Zhang. Regressor selection and wavelet network construction. Publication Interne 709, IRISA, Campus de Beaulieu-35042 Rennes Cedex - France, April 1993.