International Journal of Neural Systems, Vol. 10, No. 5 (2000) 379–395
© World Scientific Publishing Company

FEATURES SELECTION AND ARCHITECTURE OPTIMIZATION IN CONNECTIONIST SYSTEMS

MÉZIANE YACOUB and YOUNÈS BENNANI
LIPN-CNRS, Institut Galilée Université Paris 13,
Avenue J.B. Clément 93430 Villetaneuse, France
E-mail: {yacoub,younes}@lipn.univ-paris13.fr

Received 11 January 2000
Revised 22 September 2000
Accepted 22 September 2000

In this paper, we propose a features selection measure and an architecture optimization procedure for Multi-Layer Perceptrons (MLP). The algorithm presented in this contribution employs a heuristic measure named HVS (Heuristic for Variable Selection). This new measure allows us to identify and select important variables in the features space. This can be achieved by eliminating redundant features and those which do not contain enough relevant information. The proposed measure is used in a new procedure aimed at selecting the "best" MLP architecture given an initial structure. Application results for two generic problems: regression and discrimination, demonstrates the proposed selection algorithm's effectiveness in identifying optimized connectionist models with higher accuracy. Finally, an extension of HVS, named εHVS, is proposed for discriminative features detection and architecture optimization for Time Delay Neural Networks models (TDNN).

1. Introduction

It is well admitted that the Multi-Layer Perceptrons (MLP) are an interesting alternative to parametric modelization. Many research have recognized the potential of MLP for pattern recognition applications. One shows that for a fixed precision, there exists always a MLP that approximates a continuous function: MLPs are universal approximators.[10,11,25] The problem becomes complicated if one considers that the function to be approximated is not analytically known, but in the form of couples (input, output) whose set of observations are possibly smirched of error.

Two problems can be posed then:

- The first, that we will not approach explicitly in this paper, is the number of (input, output) couples needed to make a "good" function approximation. Corresponding works on this problem, have been undertaken in the PAC (Probably Almost Correct) learning.[22]
- The second, make the objective of this contribution, is relative to the determination of appropriate model sizes or model architecture as well as to the estimation of its parameters with a fixed size of the learning database.

Nevertheless, it is necessary to note that these two problems are not independent. Indeed, the quality of the parameters estimation depend closely on the ratio between the learning database size and the number of parameters to be estimate.

The determination of appropriate MLP structure can be done using simple approach that consists in training many networks, each with a different number of hidden cells, and then select the network with the best post training characteristics. This approach increases training time. Other somewhat more sophisticated approaches have been used (see the next section). In this paper, we propose a features selection measure and architecture optimization algorithm aimed at selecting the best MLP structure given an initial architecture.

The paper is organized as follows: the next section gives briefly a review of the existing approaches. Section 3 provides the principle bases for the new heuristic measure. Section 4 shows results on the proposed measure abilities for pertinence detection and quantification. Section 5 shows results from the HVS method for features selection on two generic problems (discrimination and regression) and compares them to similar results obtained using other approaches. Section 6 proposes an extension of HVS procedure to hidden layers aimed at selecting the best architecture for discrimination and regression problem. Section 7 gives an integrated approach named εHVS to features and architecture optimization for convolutional connectionist models. The objective is to select single features which are likely to have good discriminatory power. Section 8 contains the concluding remarks.

2. Variable Selection and Neural Networks

Generally, variable selection necessitates three essential elements:

- A pertinence measure: For a classification problem, one tests the system discrimination quality in the presence or in the absence of a given variable. On the other hand, for a regression problem one tests rather the quality of prediction as a function of variables.
- A research procedure: Most of pertinence measures used for the variables selection are not monotonous. It would be necessary to test all possible variable combinations ($2^n - 1$ subsets of n variables). It is a combinatorial problem and this complete research becomes impracticable. An alternative consists in the use of a sub-optimal solution as the sequential procedures: Forward Selection and Backward Elimination.
- A stop criterion: The optimal number of variables is *a priori* unknown, the use of a rule to control the selection/elimination of variables allows to stop the research when no variable is more sufficiently informative. A heuristic, that we will use in our study, consists in calculating for different variables subsets selected an estimation of the generalization error. The selected variable subset is that minimizes the generalization error.

We can classify all proposed methods, for variable selection, into two main categories:

- derivative based methods, and
- parameter based methods.

The first category uses the first or the second derivative to estimate weight saliency. The most widely used methods are: Optimal Brain Damage (OBD) due to Le Cun *et al.*,[14] Optimal Cell Damage (OCD) by Cibas in Ref. 5 which is an extension of OBD, the Optimal Brain Surgeon (OBS).[9] Both methods are based on systematic estimation of weight saliency which is defined as the change in training error when the particular weight is pruned. While OBD is based on the diagonal approximation of the Hessian and is used iteratively, alternating pruning followed by retraining, OBS is based on the full Hessian and retraining is not necessary. Recently, Pedersen proposed γOBD and γOBS,[18] to estimate the weight saliency as the associated change in generalization error if the weight is pruned. The use of the first derivatives for variable selection can be found for example in Refs. 7, 19, 20. The second category methods for variable selection are based on model parameters. Some of these methods use statistical tests to evaluate a confidence interval for each connection,[6] the mutual information criterion to evaluate a set of candidate features and to select an informative subset to be used as input data for a NN classifier,[1] the Bayesian technique of automatic relevance determination,[15–17] heuristic measures based on computing the effect an input has on the output of the Neural Network.[2,26–28]

3. HVS: Heuristic for Variable Selection

Let $\mathrm{A(I, H, O)}$ be a feed-forward neural network architecture with an input layer I, a hidden layer H, and an output layer O. The connection weight value w_{ij} between two neurons i and j reflects their link importance. This value can be positive or negative depending on whether the connection weight is an excitatory synapse or an inhibitory one. Since a connection weight can be an important excitatory synapse or an inhibitory one, we are interested only to the strength of the connection weights. We quantify the strength of each connection weight by the absolute value $|w_{ij}|$ of its associated weight value.

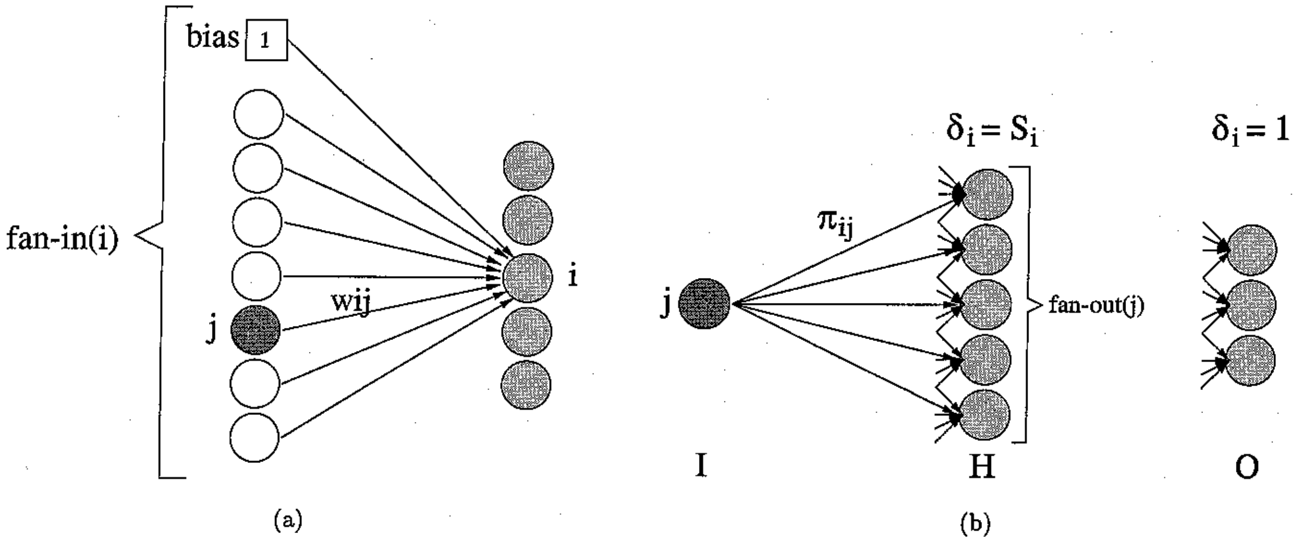

Fig. 1. (a) The partial contribution π_{ij} of a unit j connected to unit i is defined as the ratio of the absolute value of w_{ij} and the sum of the absolute values of all weights w_{ik}, $k \in$ fan-in(i). (b) The relative contribution S_j of the unit j to the output decision of the network is defined as the weighted sum $\sum \pi_{ij}\delta_i$, $i \in$ fan-out(j).

In this case, we define the partial contribution of a unit j connected to a unit i through a connection weight w_{ij} as:

$$\pi_{ij} = \frac{|w_{ij}|}{\sum_{k \in \text{fan-in}(i)} |w_{ik}|} \qquad (1)$$

where fan-in(i) is the set of unit that have a connection to unit i (see Fig. 1(a)); by definition we have: $\forall i \in I$, fan-in(i) $= \emptyset$.

The numerator of Eq. (1) measures the relation strength between unit j and unit i. The denominator is only a normalization term. The term π_{ij} defines the partial contribution of unit j connected to unit i as the proportion of all the connections strength arriving to unit i.

Now, our interest is to estimate the relative contribution of the unit j to the output decision of the network. The unit j send connections to a set of units fan-out(j) with partial contributions π_{ij}, $i \in$ fan-out(j) (see Fig. 1(b)). Each unit i in fan-out(j) has a relative contribution δ_i. We define the relative contribution of unit j to the output decision of the network as the weighted sum of its partial contributions by the relative contributions of the units to which it send connections. Thus, we obtain the following expression:

$$S_j = \sum_{i \in \text{fan-out}(j)} \pi_{ij}\delta_i \qquad (2)$$

with

$$\delta_i = \begin{cases} 1 & \text{if unit } i \in O \\ S_i & \text{if unit } i \in H \end{cases} \qquad (3)$$

An intuitive interpretation of the HVS definition can be given as follows: the partial contribution π_{ij} defined as the ratio of the absolute value of w_{ij} and the sum of the absolute values of all weights in fan-in(i) estimate the proportion of the contribution of unit j to unit i. This proportion is calculated for each connection in fan-out(j), and term δ_i is then used as a weight factor for the partial contributions to evaluate the relative contribution S_j.

Notice that HVS measure can be applied to an MLP having any number of hidden layers. In fact, we can compute partial contributions of all the units in the network using Eq. (1). Then we can compute relative contributions of all the units starting from the last hidden layer to the input layer using Eqs. (3) and (2).

3.1. *HVS for variable selection*

In this section, we describe our method for variable selection based on HVS measure.

3.1.1. *Principle of the method*

Our method for variable selection uses backward search. We eliminate the variable one by one until only one variable remains. Each time, the least important variable according to HVS measure is eliminated and the network is retrained. The algorithm for variable selection we propose is the following:

(a) Train the network using the procedure of early stopping.
(b) Compute the pertinence of each input variable according to HVS measure.
(c) Prune the least pertinent input variable.
(d) Iterate to step (a), until the last variable.

3.1.2. *Selecting the best set of variables*

The precedent algorithm generates a set of k networks (k is the number of variables) having less and less variables. Let $MLP(p)$ $p = 1, \ldots, k$ be those networks and $E(p)$ be the estimated error on validation set. Two different approaches can be used for selecting the subset of variables:

- the first approach consist to choose the network $MLP(p^*)$ which obtain the least error

$$MLP(p^*) = \arg \min_{MLP(P)} E(p)$$

and to choose the subset of p^* variables associated to $MLP(p^*)$.

- the second approach consist to use a statistical test (Fisher test) and to search among all the networks $MLP(p)$ those which are statistically close to $MLP(p^*)$. This principle allows to obtain a set of networks such that $E(p^i) \approx E(p^*)$.

  The selected subset of variables is chosen as the smallest subset of p^0 variables statistically close to $MLP(p^*)$

$$p^0 = \min_i \{p^i\}$$

For simplicity reasons, we chose to develop the first approach.

In the next section, we show two simple problems that HVS measure can be used for estimating the importance of each input variable.

4. HVS Abilities for Estimating Importance of Input Variables

In this section we want to verify whether HVS measure can be used for ranking the input variables according to their importance. First, we consider a simple problem to test whether this measure is capable of identifying the significant variables from the input signal, then we consider another problem to verify whether it can estimate the ratio between important variables.

For all problems considered in this paper, we use a three layered fully connected MLP. All the networks are trained using the back-propagation algorithm. We will use the following notation to represent the neural networks' architecture: $\langle n_I | n_H | n_O \rangle$, where n_I represents the dimension of the input layer, n_H the number of hidden neurons, and n_O the number of neurons in the output layer.

4.1. *Pertinence detection*

In order to test whether HVS measure can detect important variables from the input signal, we consider the "even-Parity-3" problem. In this problem (Fig. 2), the data set contains two classes, positive examples and negative examples. Each input example is composed of n binary variables. An input vector $(x_1, x_2, x_3, \ldots, x_n)$ is a positive example if and only if an even number of its first three bits x_1, x_2, and x_3 are on. In our experiments $n = 10$. In this problem only the first three bits are important and are of equal importance.

We generated a data set of 2956 instances: $\frac{2}{3}$ of the data set are used for learning and the $\frac{1}{3}$ remaining instances are used for validation. We have used an architecture $\langle 10|2|2 \rangle$. At the end of learning, we used HVS measure to estimate the importance of each input variable. We present in Fig. 3 the results by drawing the importance of each input variable according to HVS measure. HVS measure indicates precisely that there are two classes: one contains input variables 1, 2, and 3 which are of

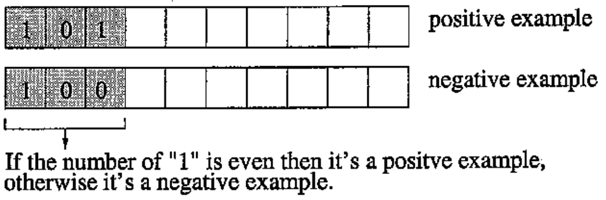
1 0 1
positive example
1 0 0
negative example
If the number of "1" is even then it's a positve example, otherwise it's a negative example.

Fig. 2. "Even-parity-3" problem: only the first three bits are important.

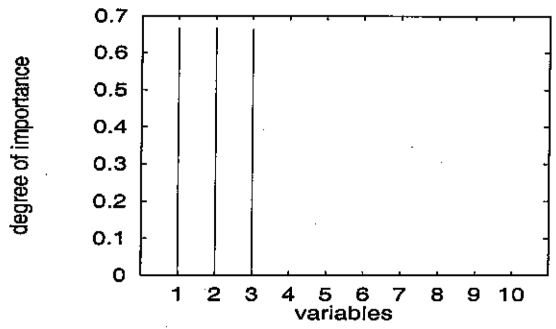

Fig. 3. Degree of Importance: Even parity problem.

approximately equal importance and the other contains the remaining input variables which are not important.

4.2. *Pertinence quantification*

In order to test whether HVS measure can estimate the ratio between important variables, we consider the "multiplexer 11" problem. In this problem (Fig. 4), each instance is a boolean vector divided into 3 address bits and 2^3 data bits. The first three bits are the address bits: they allow to address the remaining eight bits. If the addressed bit has the value equal to 1, the instance is a positive example, otherwise it is a negative example. Consequently, the first three bits are of equal importance and are, theoretically, eight times more important than the remaining bits which are also of equal importance.

We used a data set of 2048 instances: $\frac{2}{3}$ of the data set are used for learning and the $\frac{1}{3}$ remaining instances are used for validation. We have used an architecture $\langle 10|8|2 \rangle$. At the end of learning, we used

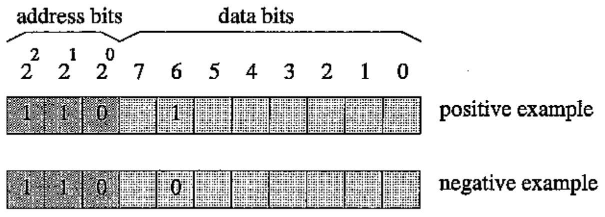

Fig. 4. "Multiplexer11" problem: the first three bits define the addressed bit. In this example it's the 6th bit: $(1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)$. The instance is positive if the 6th bit is equal to 1, and it's negative if the 6th bit is equal to 0.

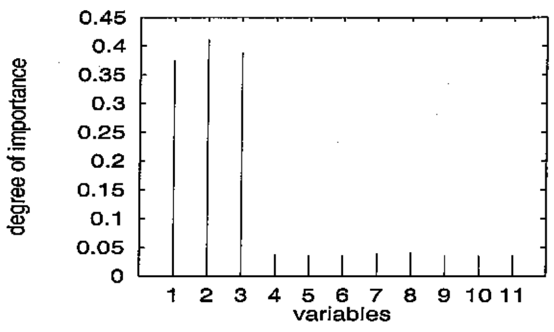

Fig. 5. Degree of Importance: Multiplexer11 problem.

HVS measure to estimate the importance of each input variable. We present in Fig. 5, the results by drawing the importance of each input variable according to HVS measure. HVS measure indicates precisely that there are two classes: one contains input variables 1, 2, and 3 which are of approximately equal importance and the other contains the remaining input variables which are also of approximately equal importance. Notice that the ratio between the importance of the first and the second class is almost equal to the theoretical ratio.

5. Features Selection Using HVS Procedure

We now present two validation problems: the Breiman "waveforms" discrimination problem and the well known Sunspot regression problem.

5.1. *Discrimination task*

To show the performances of our method on discrimination problem, we choose the "Waveform" classification problem. It was proposed by Breiman *et al.*[4] and a noisy version was used by De Bollivier *et al.*[3] It is a three class problem. The examples are real vectors of dimension 40 including 19 noise components. The representation of the Waveform problem in the two first principal components space is shown in Fig. 6. Polygons show the hull for each class. Notice the high over-loping between all classes.

For this problem, our experimental design is as follows: for learning we use a set D^l of 300 instances and a set D^v of 700 instances for validation. For test, we use a set D^t of 4300 instances. We have used the initial architecture $\langle 40|10|3 \rangle$.

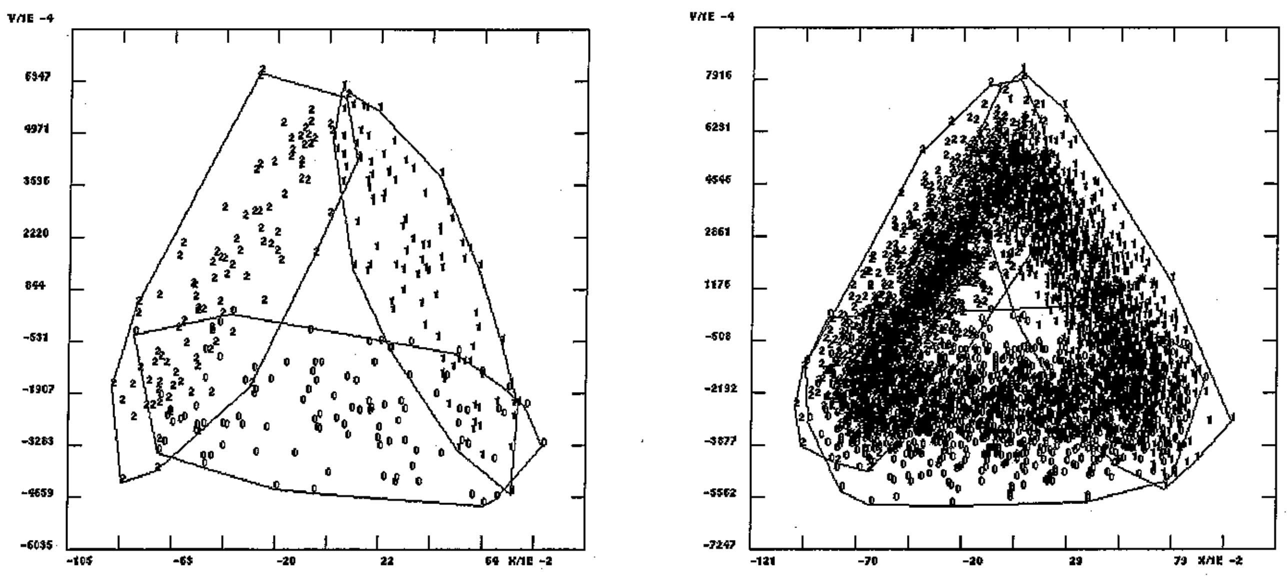
Fig. 6. Representation of the Waveform problem in the two first principal components space for the learning (left) and test set (right). Polygons show the hull for each class.

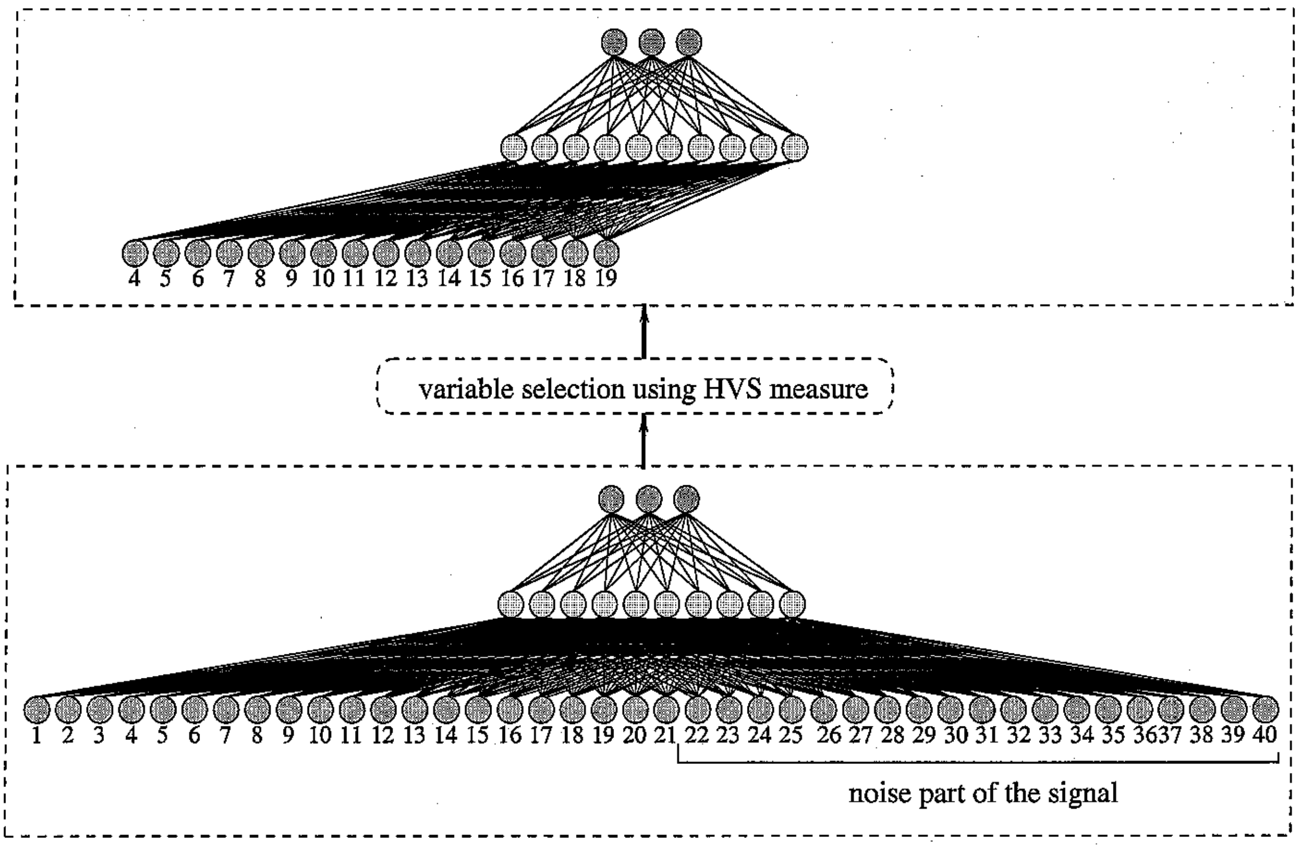

Fig. 7. Selected variables using HVS measure.

Table 1. Performances before and after variable selection using HVS measure.

	Performances on		
	D^l	D^v	D^t
Before variable selection	92.00 [88.37, 94.57]	83.85 [80.95, 86.40]	82.83 [81.68, 83.93]
After variable selection	88.66 [84.58, 91.78]	85.57 [82.77, 87.98]	85.37 [84.28, 86.40]

As shown in Fig. 7, HVS procedure has selected the subset of variables

$$S^v(\text{HVS}) = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\},$$

and rejected variables 1 to 3 and 20 to 40. It may be noticed that the HVS procedure selected the most important variables in the first part of the signal, and rejected all variables in the "noise" part of the signal.

As shown in Table 1, after variables selection using HVS measure, performances on D^t are increased from 82.83 to 85.37. Notice that the result of our approach is close to the Bayesian theoretical limit which is approximately equal to 14% error's rate on D^t.

5.2. *Regression task*

To show the effectiveness of our method on a regression task, we choose a real-world time series of limited record length, the sunspot data. It contains the annual averaged sunspot activities from 1700 to 1979 (280 points). The series is shown in Fig. 8.

For this problem, we attempt to predict one value using the 12 previous ones, leading to a network with 12 input units and one output unit. We use the data from 1712 to 1920 for learning (a set D^l of 209 examples) and the data from 1921 to 1955 for validation (a set D^v of 35 examples). For the test, we use the data from 1956 to 1979 (a set D^t of 24 examples). We have used the initial architecture $\langle 12|10|1\rangle$ with the hyperbolic tangent and identity as transfer functions of the hidden and output layer, respectively. We start with 12 inputs neurons, because Weigend *et al.*[24] notice that increase from 12 to 25 input units did not lead to significant further improvement.

Performance is measured in terms of average relative variance (arv), which is the ratio between the average squared error of the model and the variance of the data. Notice that the definition of the arv quantity would relate the error of the model and the variance of the data calculated on the same set D (taken from the entire Time Series TS).

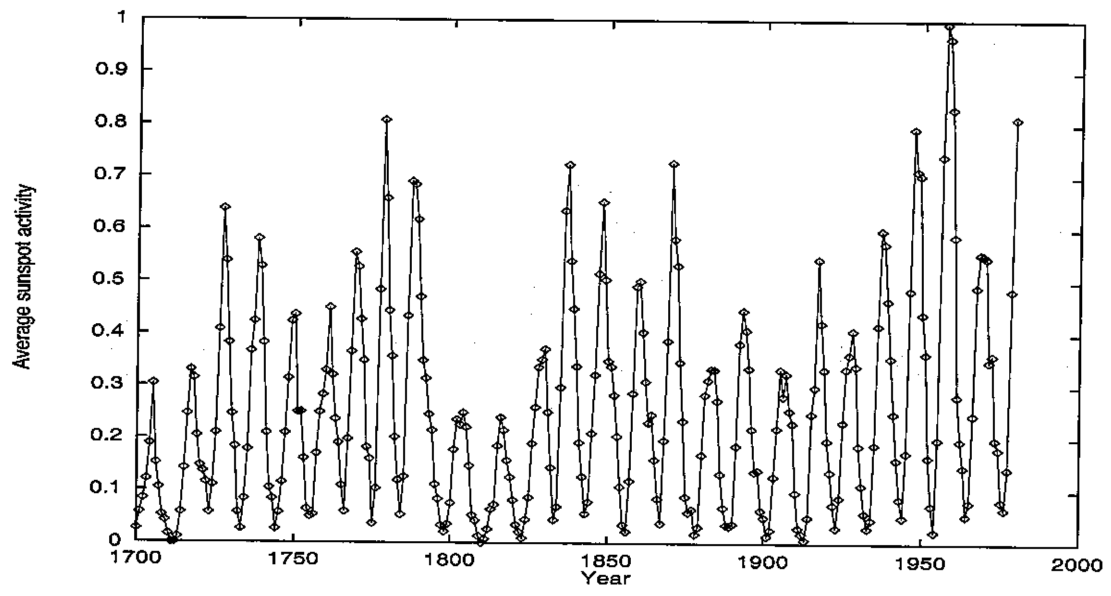

Fig. 8. The sunspots data for years 1700 to 1979.

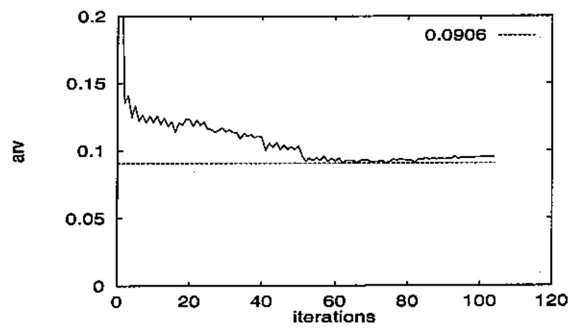

Fig. 9. Arv evolution on D^v during learning phase.

$$\mathrm{arv}(D) = \frac{\sum_{i\in D}(y^{(i)} - f(x^{(i)}))^2}{\sum_{i\in D}(y^{(i)} - \mu_D)^2}$$

where $y^{(i)}$ is the true value of the time series at time i, $f(x^{(i)})$ is the output of the network at time i, and μ_D the average of the target value in D.

However, the widely used definition of the arv scales by the overall variance of the data.

$$\mathrm{arv}(D) = \frac{\frac{1}{|D|}\sum_{i\in D}(y^{(i)} - f(x^{(i)}))^2}{\frac{1}{|S|}\sum_{i\in TS}(y^{(i)} - \mu_{TS})^2}$$

In order to compare our arv with established results, we will stick to the widely used definition.

Now we illustrate our method for variable selection in more details through one experiment. Figure 9 shows the arv evolution on D^v during learning phase. The arv on D^v at the end of learning is equal to 0.0906.

At the end of learning phase, we proceed to variable selection. We present on Fig. 10 the arv evolution on D^v during variable selection phase.

To select the best system (the best subset of input variables), we used a "global early stopping". In other words, we choose the system which obtained the best arv on D^v. Figure 11 shows the best arv on D^v obtained by each system, each vertical line represents one system. System i corresponds to the system obtained after pruning i least important variables according to HVS measure. As shown by this figure, the best system is obtained after pruning six

Table 2. We have done 30 experiments, the delays x_{t-12}, x_{t-11}, x_{t-8}, x_{t-3}, x_{t-2}, and x_{t-1} have been selected in the majority of these experiments. Reported in this table are our results averaged over 30 experiments after variables selection.

D^l(1712–1920)	D^v(1921–1955)	D^t(1956–1979)
0.089	0.076	0.331

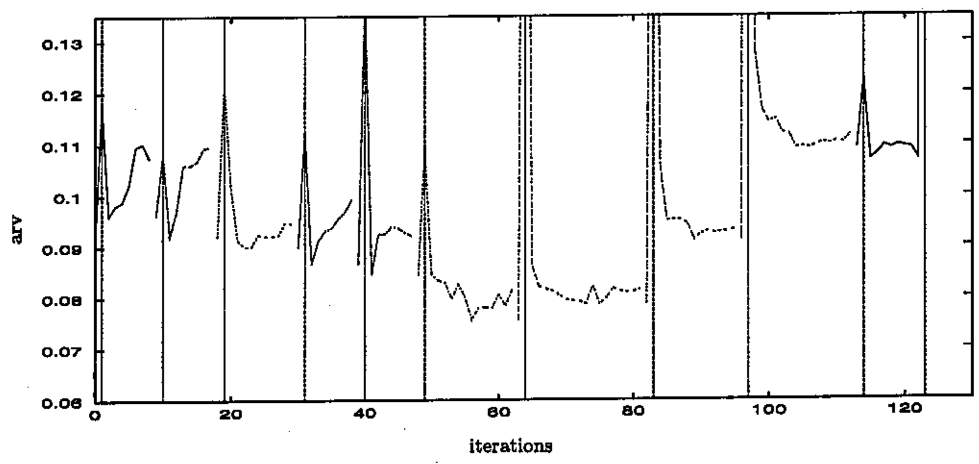

Fig. 10. Arv evolution on D^v during input variable selection procedure. The first point in this figure (iteration 0) corresponds to the best arv obtained during learning phase (0.0906). Each vertical line corresponds to the iteration where an input variable is pruned. Each Interval delimited by vertical lines corresponds to retraining phase after pruning input variable. The point before each vertical line corresponds to the best arv on D^v obtained during the retraining phase.

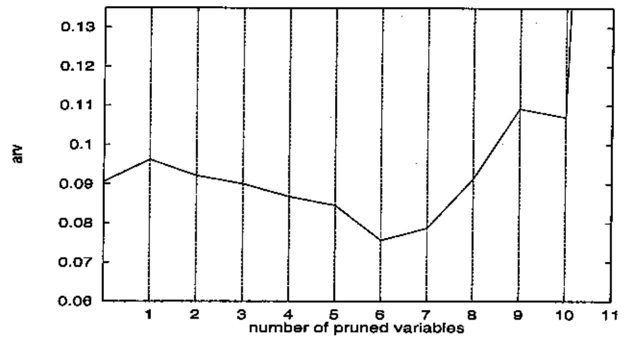

Fig. 11. Global arv evolution on D^v during input variable selection procedure. The arv after eliminating i variables corresponds to the best arv obtained during the ith retraining phase in Fig. 10.

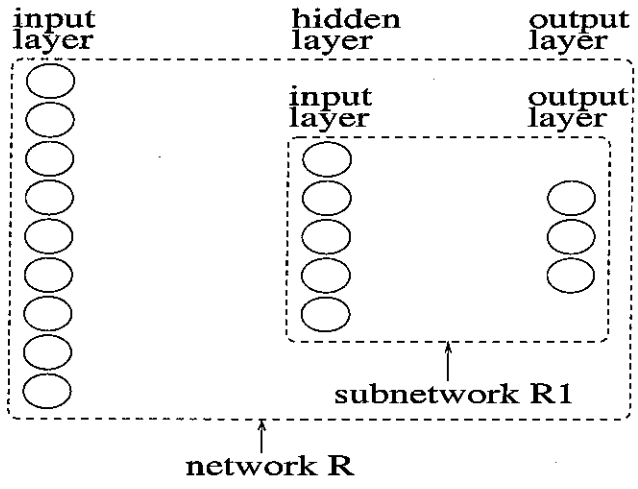

Fig. 12. The hidden layer of the network R is considered as the input layer of the subnetwork R1.

variables. The subset of selected variables are x_{t-12}, x_{t-11}, x_{t-8}, x_{t-3}, x_{t-2}, and x_{t-1}. Reported in Table 2 are our results averaged over 30 experiments after variables selection.

6. HVS for Architecture Optimization

In this section, we show that HVS measure can be used for architecture optimization.

6.1. *Principle of the method*

Our method for architecture optimization consists in extending the application of HVS measure to the hidden units by viewing each hidden layer as an input layer of a subnetwork made of this layer, the output layer, and all the layers that are between these two layers (see Fig. 12). Thus the problem of architecture optimization became a problem of variable selection in the subnetwork considered.

6.2. *Experimental validation*

In order to illustrate the effect of using HVS measure for hidden units pruning, we continue our experiments on the Breiman "waveforms" discrimination problem and the Sunspots regression problem, described in Sec 5.

6.2.1. *Breiman "waveforms" discrimination problem*

After variable selection, we used HVS measure for hidden units selection. We present on Fig. 13 the

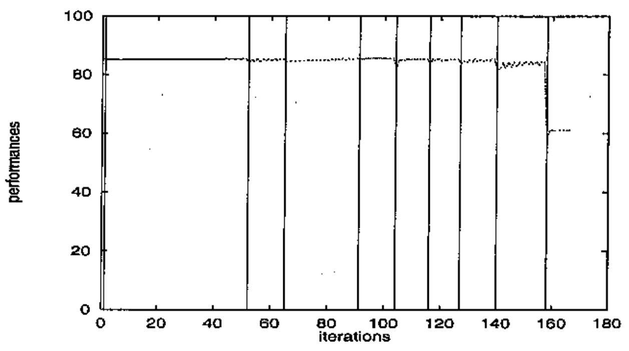

Fig. 13. Performances evolution during hidden neurons pruning on D^v.

results by drawing the performances on D^v during hidden neurons pruning. The best performance on D^v is obtained after pruning four hidden neurons, performances are increased from 85.57 to 85.71.

As shown in Table 3, using HVS for input and hidden neurons pruning allows to cut 320 parameters, which corresponds to a percentage of deletion of 72.23%, and performances are increased from 82.83 to 85.46 on D^t.

6.2.2. *Sunspots regression problem*

After input units pruning we proceed to pruning hidden units, this process can be seen as a hidden dimensions discovering process: the number of hidden units can be interpreted as an upper limit of the dimension of the dynamics of the time series.

Table 3. Performances and number of parameters before pruning, after variable selection, and after variable selection and hidden neurons pruning.

	Architecture	Performances on D^l	D^v	D^t
Before pruning	⟨40|10|3⟩ (443 parameters)	92.00 [88.37, 94.57]	83.85 [80.95, 86.40]	82.83 [81.68, 83.93]
After variable selection	⟨16|10|3⟩ (203 parameters)	88.66 [84.58, 91.78]	85.57 [82.77, 87.98]	85.37 [84.28, 86.40]
After variable selection and hidden neurons pruning	⟨16|6|3⟩ (123 parameters)	89 [84.95, 92.06]	85.71 [82.93, 88.11]	85.46 [84.38, 86.49]

We present in Fig. 14 the arv evolution on D^v during hidden neurons pruning for the experiment we have described in Sec. 5.2. The best arv is obtained after pruning seven hidden neurons. As shown in Fig. 15 using HVS measure to input variables and hidden neurons pruning allows to cut 116 parameters, which corresponds to a percentage of deletion of 82.27%. Table 4 shows, for this experiment, the results obtained after variable selection and hidden neurons pruning.

We have applied our architecture optimization procedure to each of the 30 experiments done in Sec. 5.2. Reported in Table 5 are our results, averaged over 30 experiments, after input variables selection and pruning hidden neurons, and some results obtained using different methods. Notice that our method allows not only to reduce the effective complexity of the model but also to obtain predictions with better precisions compared to the prediction qualities of the other approaches.

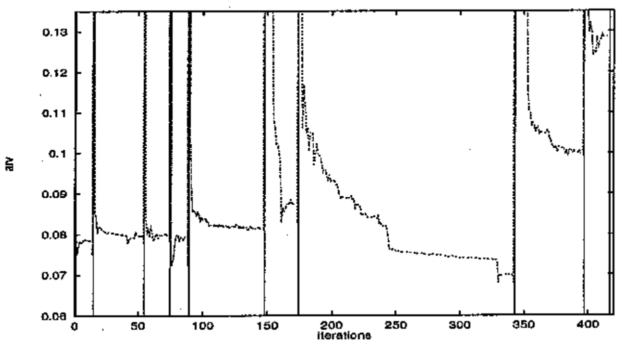

Fig. 14. Arv evolution on D^v during hidden units pruning.

Table 4. Results for one experiment.

	D^l	D^v	D^t
At the end of learning phase	0.0832	0.0906	0.4032
After variables selection	0.0802	0.0756	0.3437
After variables selection and hidden neurons pruning	0.0869	0.068	0.2968

7. Discriminative Features Extraction and Selection

In this section, the focus is not to build a face recognition system, but to introduce a feature selection approach to detect and identify discriminative zones in the retina (input layer). This extraction is based on minimizing a classifier's errors through a discriminative training. This approach allows to select single features which are likely to have good discriminatory power and extract nonlinear combinations of features with the same aim. The proposed method εHVS, is based on the HVS measure. HVS has shown successful results on Multilayer Perceptron Networks for variable selection (see Sec. 5)[26] and for architecture optimization (see Sec. 6).[27] In this section, we are interested in the convolutional connectionist models named TDNN (Time Delay Neural Networks).[12] These networks are redundant and incorporate *a priori* knowledge to achieve partial invariance to translation, rotation, scale, and deformation. The goal is to use εHVS method to reduce redundant informations in the TDNN networks. In other words, reducing redundant informations means optimizing a priori knowledge used to build the initial model.

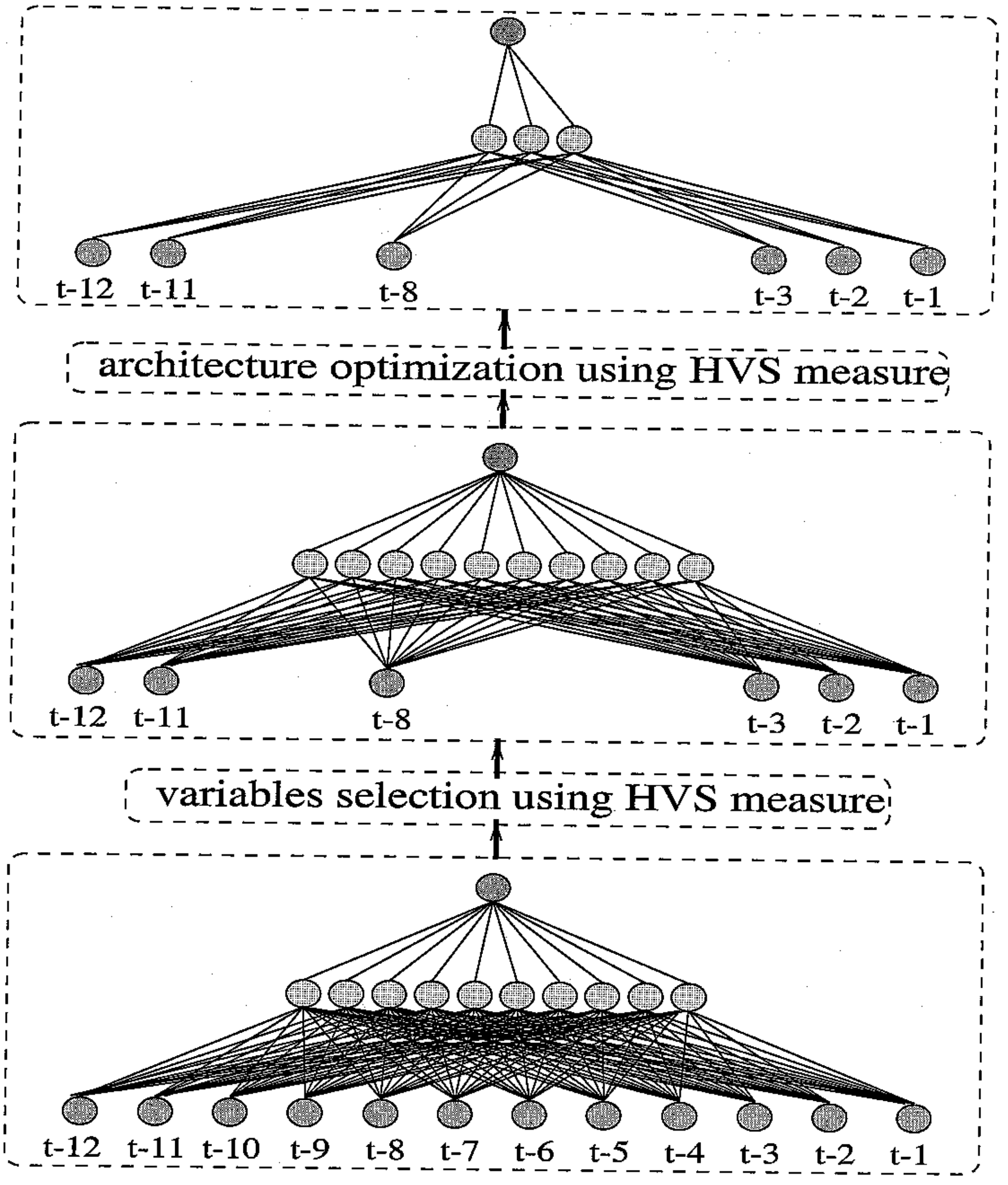
t-12 t-11 t-8 t-3 t-2 t-1
architecture optimization using HVS measure
t-12 t-11 t-8 t-3 t-2 t-1
variables selection using HVS measure
t-12 t-11 t-10 t-9 t-8 t-7 t-6 t-5 t-4 t-3 t-2 t-1

Fig. 15. The effect of using HVS measure for input and hidden neurons pruning.

The first part of the proposed approach is concerned with the classifier optimization jointly with the feature extractor. It is shown how this optimization can be viewed as a variable selection problem. The second part propagates this effect of the first one to the input layer of the system. This propagation allows us to identify features which are likely to have good discriminatory power for the task. The proposed method is applied to face recognition. This problem is complex. For an overview of face recognition, see Ref. 23. We present a successful demonstration on convolutional connectionist system's application to face recognition task with varying facial detail, expression, pose, etc. The resulting features show that eyes, nose, mouth, chin, and hair regions are important to the discrimination task. These

Table 5. Comparison of the results obtained using different methods.

Model/arv	D^l(1712–1920)	D^v(1921–1955)	D^t(1956–1979)
Yacoub & Bennani	0.090	0.071	0.318
Goutte[8]	0.082	0.082	0.357
Svar *et al.*[21]	0.090	0.082	0.35
Weigend *et al.*[24]	0.082	0.086	0.35
Linear	0.131	0.128	0.36

features are compared to standard knowledge-based features concerning the task. Our approach and the knowledge-based approach can influence each other: the knowledge-based feature can be used as a starting partition for optimization and analysis of the selected features can provide new insight on the task.

7.1. *Description of data set*

Our database set consists of four different persons. Ninety-two pictures have been taken under different angles (23 images per person), they are digitized in a 40×50 gray shades. No other preprocessing is performed on the data: they are simply put into a large 2000 cells input layer.

7.2. *System architecture*

The network we used is shown in Fig. 16. The system has two modules: the first one is a feature extractor with two shared weights-hidden layers producing at the last hidden layer a feature map of the image presented at the input; the second one is a classifier working on the feature map. The output of the classifier gives the identity of the presented image. The output layer has as many units as there are persons to identify (four units in our case) and is fully connected to the last hidden layer.

In this model, we use shared weights connection for the first module: all units in one layer are locally connected to their inputs through a receptive field of the same size, and their weights are identical. This connection schema is similar to having a filter passed through the input layer, the coefficient of the filter being the network weights.

7.3. *εHVS principle*

The method we propose for selecting the discriminative regions consists in three steps: features extraction, features selection, and discriminative zones identification.

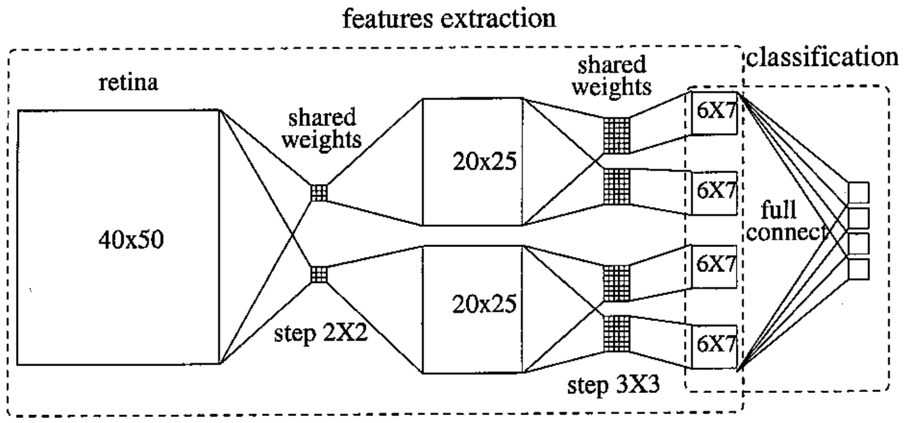

Fig. 16. Architecture for face identification. The system has two successive module: one performs features extraction and the other classification. This kind of architecture takes advantage of the 2D structure of the input pattern.

Step 1: Features extraction

This step consists to do learning using the two modules, it allows the first module to extract features. These features are not trivial and we cannot understand their precise nature. The important thing is that these features are not all informatives: most of them are redundant indeed even not pertinent. Thus in the next step, we propose to select the most pertinent features from those produced by the feature. extractor module.

Step 2: Features selection

To select pertinent features from those proposed by the first module, we apply our variable selection procedure[26,27] to the second module. The main goal of this step is to select a subset of n features from the given set of N measurements ($n \ll N$), without significantly degrading the performance of the recognition system.

It is important to notice that to evaluate the importance of the features extracted by the first module, HVS measure uses only the parameters of the second module. Thus, features selection using HVS measure does not require many computations relatively to this kind of architecture.

Step 3: Discriminative zones identification

To select discriminative zones of the input, it suffice to retropropagate the effect of the variable selection method on the first module. The principle is very easy (see Fig. 17):

$$\text{if fan-out } (i) = \emptyset \text{ then}$$
$$\forall j \in \text{fan-in}(i),\ w_{ij} = 0\,.$$

We apply this principle starting from the last hidden layer (the last layer of the features extractor module). Notice that this step does not require any computation, we do only test on the fan-out(i) where i is a unit. The set of the selected units on the input represente the discriminative zones.

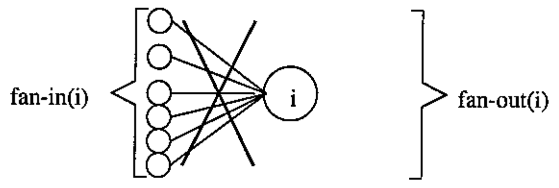

Fig. 17. For all unit i, if fan-out(i) = $\emptyset$ then we prune each connection w_{ij}, $j \in$ fan-in(i).

7.4. *Experimental validation*

A principle difficulty in face recognition comes from the appearance variation of the face due to facial expressions. The proposed approach has two corresponding steps: identifying discriminating and invariant local features based on hidden layers coding, and selecting discriminative zones in the retina of the system.

We have used a very redundant architecture (Fig. 16). The face to be identified is presented to the input layer of 40×50 units. The first hidden layer has two clusters of size 20×25 connected through shared weights to the input layer (receptive fields are 3×3 and overlap by 2 pixels). The second hidden layer has four clusters of size 6×7. Each cluster of the first hidden layer is connected through shared weights to two clusters of the second hidden layer (receptive fields are 5×7 and overlap by 2). The second hidden layer is fully connected to the output layer (with four cells, one for each person). Table 6 shows the number of weights and number of connections for each module.

We have used a learning set D^l of 76 examples and a set D^v of 16 examples. At the end of learning performances are 100% on D^l and 100% on D^v. Figure 18 shows the network state before features selection, when presenting at the input a face of Guy 3.

We applied to the classifier module our variable selection method. Table 7 shows performances and the number of features used by the classifier before and after variable selection. Notice that we have pruned 158 features, which corresponds to a percentage of deletion of 94%, while keeping the same

Table 6. Number of weights and number of connections of each module of the identification system.

	Features extractor module	Classifier module
Number of connections	14880	672
Number of weights (free)	158	672

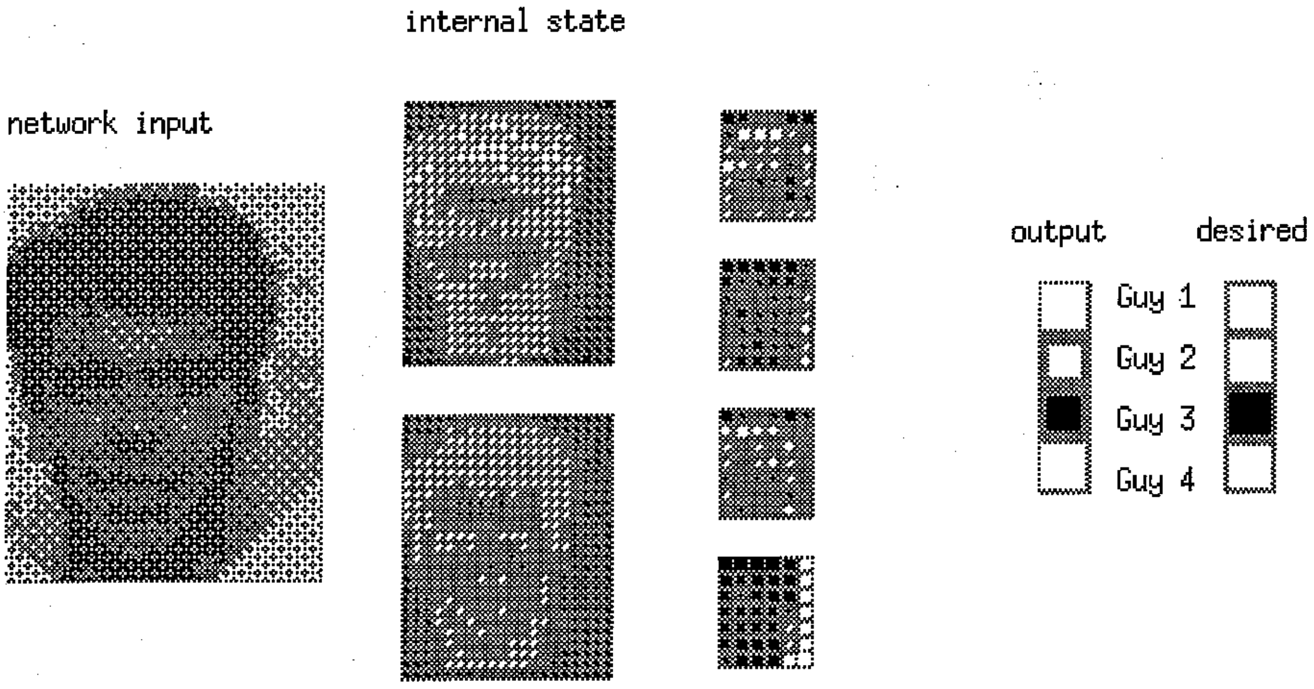

Fig. 18. Network state before features selection, when presenting at the input a face of Guy 3. The second module uses 168 features to identify the image presented to the input.

Table 7. Among the 168 features extracted by the first module, only 10 have been selected using HVS measure, while still keeping performances at 100% on D^l and on D^v.

	Classifier		Performances	
	# features	# param.	D^l	D^v
Before selection	168	672	100% [95.19, 100]	100% [80.64, 100]
After selection	10	40	100% [95.19, 100]	100% [80.64, 100]

Table 8. HVS effect on the first module units and on the system performances.

	Number of cells		Performances	
	Input layer	First hidden layer	D^l	D^v
Before selection	2000	1000	100% [95.19, 100]	100% [80.64, 100]
After selection	1077	482	100% [95.19, 100]	100% [80.64, 100]
Percentage of deletion	46%	51.8%		

performances on D^l and D^v. This deletion allows us also to reduce the number of parameters of the classifier, from 672 parameters to 40 parameters.

For discriminative zones identification, first we have retropropagated the effect of HVS on the first hidden layer. We have pruned 518 cells, which corresponds to a percentage of deletion of 51.8%. Next, we retropropaged HVS effect on the input layer. We pruned 923 units, which corresponds to a percentage of deletion of 46%. This is illustrated in Fig. 19.

Table 8 illustrates HVS effect for discriminative zones selection. We pruned almost half unit of the input layer while still keeping performances at 100% on D^l and 100% on D^v.

Figure 20 shows some images after discriminative zones selection using εHVS method. Despite the small size of the database, we can interpret the obtained results. The images of Fig. 20 show that

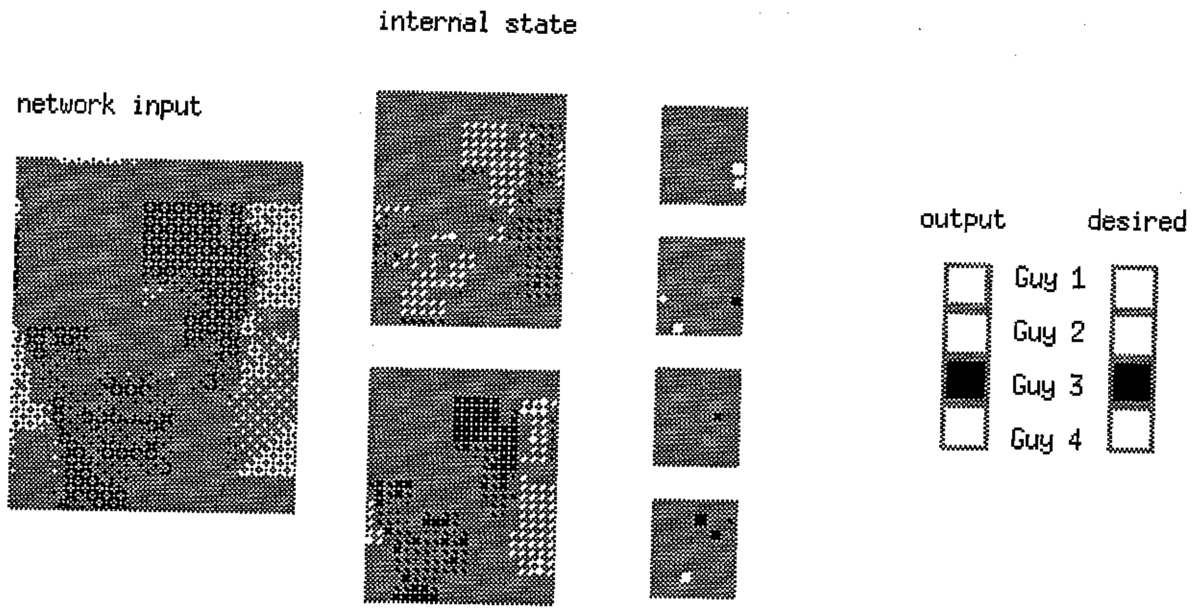

Fig. 19. HVS effect on the input and hidden units, gray regions correspond to the pruned units.

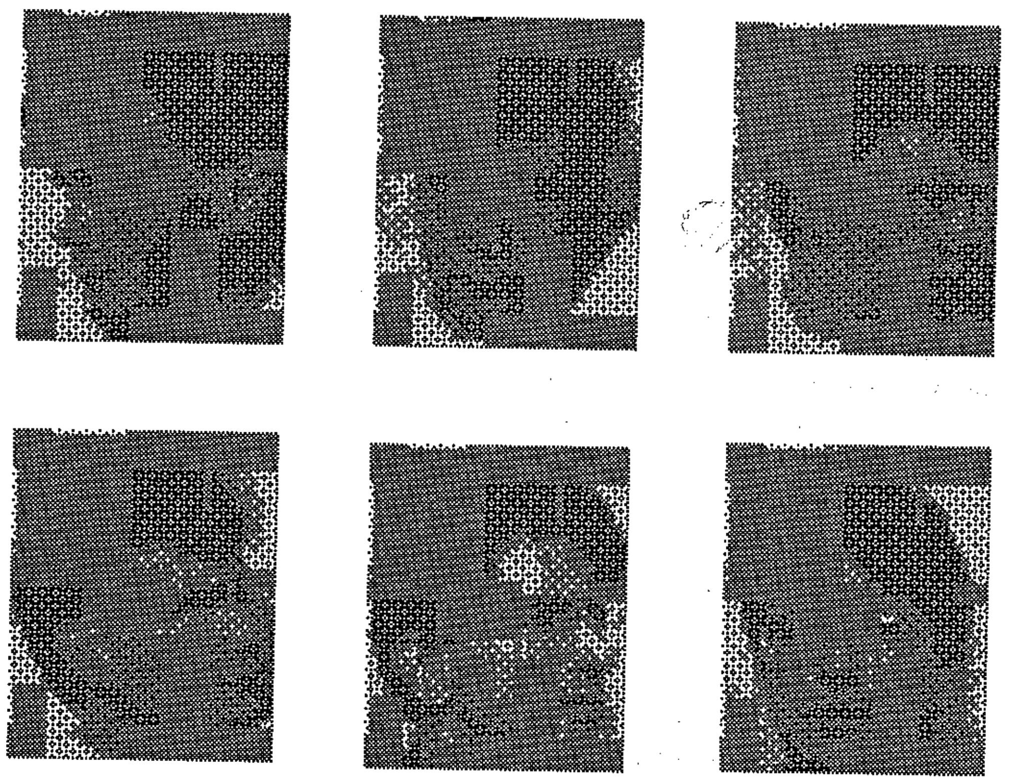

Fig. 20. Some images after pruning 46% of input units using HVS measure.

the selected discriminatory zones contain in most cases the regions of the face supposed important for the classification: eyes, mouth, nose, chin, and hair. This result confirms observations reported by other researchers, such as Lawrence *et al.*,[13] showing that for face identification, eyes, mouth, nose, chin, and hair are the most important regions. The validation of these results on a large database is under study.

Obviously, there is no reason that the discriminatory zones selected by εHVS correspond necessarily to the characteristics advised in the domain. But we can envisage a bi-directional cooperation between experts' information and results of our approach. The characteristics proposed by experts can be used as a starting point of our method. Another cooperation consists of using our results to complete or even to improve experts' results.

8. Conclusion

First, we presented our pertinence measure, called HVS, which allows to estimate the importance of each input variable of the model. HVS measure is simple and cheap to implement, we showed its abilities on two problems for which the theoretical importance of each variable is known. Then we presented a method, based on HVS measure, for input variable selection. We extended the application of HVS measure for architecture optimization by pruning irrelevant hidden units. Experimental results on one relatively hard discrimination problem and one real world time series of limited length show that using HVS measure for variable selection and architecture optimization allows not only to obtain very good results of prediction or discrimination in comparison with other results obtained by other approaches, but also to reduce considerably the effective complexity of the model. Finally, we have proposed a pruning-based method applied to a priori knowledge-based system, to optimize adaptively during a discriminative training the use of this knowledge. This optimization was performed through a features extraction and selection process. Experimental results on a complex problem show the effectiveness of this method. We are currently investigating the use of this method for identification and classification of under water acoustic signals. This is a difficult problem because of low SNRs and a high degree of variability in signals emanated from the same type of sound source.

References

1. R. Battiti 1994, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks* **5**(4), 537–550.
2. Y. Bennani and F. Bossaert 1995, "A neural network based variable selector," *ANNIE* **5**, 425–430.
3. M. de Bollivier, P. Gallinari and S. Thiria 1991, "Cooperation of neural nets and task decomposition," *IJCNN'91* **2**, 573–576.
4. L. Breiman, J. Freidman, R. Olshen and C. Stone 1984, *Classification and Regression Trees* (Wadsworth Int. Group).
5. T. Cibas, F. Fogelman-Soulie, P. Gallinari and S. Raudys 1994, "Variable selection with optimal cell damage," *ICANN'94* **1**, 727–730.
6. M. Cottrel, B. Girard, Y. Girard, M. Mangeas and C. Muller 1995, "Neural modeling for time series: A statistical stepwise method for weight elimination," *IEEE Transactions on Neural Networks* **6**(6), 1355–1364.
7. T. Czernichow and A. Munoz 1995, "Variable selection through statistical sensitivity analysis: Application to feedforward and recurrent neural networks," *Technical Report.*
8. C. Goutte 1996, "On the use of pruning prior for neural networks," *Neural Network for Signal Processing* **6**, 52–61.
9. B. Hassibi and D. G. Stork 1993, "Second order derivatives for networks pruning: Optimal brain surgeon," *Advances in Neural Information Processing Systems* **5**, 164–171.
10. R. Hecht-Nielsen 1987, "Kolmogorov's mapping neural network existence theorem," *Proc. of the IEEE 1st. Conf. on Neural Networks* **3**, 11–14.
11. K. Hornik, M. Stinchcombe and H. White 1989, "Multilayer feedformard networks are universal approximators," *Neural Networks* **2**, 359–366.
12. K. J. Lang, A. H. Waibel and G. E. Hinton 1990, "A time-delay neural network architecture for isolated word recognition," *Neural Networks* **3**(1), 23–44.
13. S. Lawrence, C. Lee Giles, A. C. Tsoi and A. D. Back 1997, "Face recognition: A convolutional neural network approach," *IEEE Transactions on Neural Networks* **8**(1), 98–113.
14. Y. Le Cun, J. S. Denker and S. A. Solla 1990, "Optimal brain damage," *Advances in Neural Information Processing Systems* **2**, 598–605.
15. D. J. C. MacKay 1994, "Bayesian methods for backpropagation networks," in *Models of Neural Networks III*, chapter 6, eds. E. Domany, J. L. van Hemmen and K. Schulten (Springer-Verlag, New York).

16. D. J. C. MacKay 1995, "Bayesian nonlinear modelling for the 1993 energy prediction competition," in *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, ed. Heidbreder G. (Kluwer, Dordrecht).
17. R. M. Neal 1994, "Bayesian learning for neural networks," Ph.D. thesis, University of Toronto, Canada.
18. M. W. Pedersen, L. K. Hansen and J. Larsen 1996, "Pruning with generalisation based weight saliencies: γOBD, γOBS," *Neural Information Processing Systems* **8**, 521–528.
19. F. Rossi 1996, "Attribute suppression with multilayer perceptron," *Proceedings of IEEE IMACS'96.*
20. D. W. Ruck, S. K. Rogers and M. Kabrisky 1990, "Feature selection using a multi-layer perceptron," *Neural Network Comput.* **2**(2), 40–48.
21. C. Svarer, L. K. Hansen and J. Larsen 1993, "On design and evaluation of tapped-delay neural networks architectures," *IEEE International Conference on Neural Networks*, pp. 46–51.
22. L. G. Valiant 1984, "Theory of the learnable," *Communication of ACM* **27**, 1134–1142.
23. H. Wechsler, P. J. Phillips, V. Bruce and F. Fogelman 1998, "Face recognition: From theory to application," *NATO ASI series* **163**.
24. A. S. Weigend, B. A. Huberman and D. E. Rumelhart 1990, "Predicting the future: A connectionist approach," *Int. Journal of Neural Systems* **1**(3), 193–209.
25. H. White 1990, "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks* **3**, 535–549.
26. M. Yacoub and Y. Bennani 1997, "HVS: A heuristic for variable selection in multilayer artificial neural network classifier," *Intelligent Engineering Systems Through Artificial Neural Networks* **7**, 527–532.
27. M. Yacoub and Y. Bennani 1998, "Architecture optimization in feedforward connectionist models," *IEEE International Conference on Artificial Neural Networks*, pp. 881–886.
28. M. Yacoub and Y. Bennani 1999, "Discriminative feature extraction and selection applied to face recognition," *International Joint Conference on Neural Networks.*