ELSEVIER

Neurocomputing 12 (1996) 223-248

NEUROCOMPUTING

Variable selection with neural networks [1]

Tautvydas Cibas [a], Françoise Fogelman Soulié [b], Patrick Gallinari [c,*], Sarunas Raudys [d]

[a] *LRI, bât. 490, Université de Paris-Sud, F-91405 Orsay, France*
[b] *SLIGOS, 1 avenue Newton, bp 207, F-92 142 Clamart cedex, France*
[c] *LAFORIA-IBP, Univ. Paris 6, 4 Place Jussieu, F-75252 Paris cedex 05, France*
[d] *Dep. Data Analysis, Inst. Math. Informatics, Akademijos'4, Vilnius 2600, Lithuania*

Received 15 November 1994; accepted 2 October 1995

Abstract

In this paper, we present 3 different neural network-based methods to perform *variable selection*. OCD – Optimal Cell Damage – is a pruning method, which evaluates the usefulness of a variable and prunes the least useful ones (it is related to the Optimal Brain Damage method of Le Cun et al.). Regularization theory proposes to constrain estimators by adding a term to the cost function used to train a neural network. In the Bayesian framework, this additional term can be interpreted as the log prior to the weights distribution. We propose to use two priors (a Gaussian and a Gaussian mixture) and show that this regularization approach allows to select efficient subsets of variables. Our methods are compared to conventional statistical selection procedures and are shown to significantly improve on that.

Keywords: Variable selection; Regularization; Neural network pruning, Dimensionality reduction

1. Introduction

Neural Networks – NNs – are used in quite a variety of real-world applications, where one can usually measure a potentially large number P of variables X_i. Usually not all X_i are equally informative: there may be noisy components, some X_i maybe irrelevant to the problem or redundant when correlated. For small data

* Corresponding author. Email: gallinari@laforia.ibp.fr
[1] This paper extends two papers presented at the ICANN'94 conference in Sorrento: [10,11].

sets, better performances may be obtained by discarding even informative variables. In many practical applications, if one could select $p \ll P$ 'best' variables X_i, then one could reduce the amount of data to gather and process while possibly increasing performances. *Variable selection* is thus an important issue in NNs. It is also a complex problem; one needs a criterion to measure the importance of a variables subset and that value will, of course, depend on the predictor or classifier further used: a subset of variables could be optimal for one system, and very inefficient for another; an optimal subset of size p might not contain all variables of a smaller subset (non-*monotonicity*). Conventional variable selection techniques are based upon statistical or heuristic tools [15,16,36]. The major difficulty comes from the intrinsic combinatorics of the problem with the consequence that only approximate methods, based on heuristic measures of variable importance, can be used for large size problems. Most often, selection and data processing, e.g. classification, are treated sequentially and the adequacy of the selection criterion to the classifier is up to the user. Using NNs for variable selection is attractive, since they have the potential for simultaneously performing classification or approximation and variable selection: variables will thus be selected so as to optimize the training criterion.

In this paper we will present two methods for variables selection through NNs. Both are sub-optimal since they rely on heuristics, they are aimed at practical applications.

The first one is based on the evaluation of a variable usefulness. Various methods have been proposed to assess the value of a weight [29,20,22]. Using ideas similar to [22], we derive a method, called *Optimal Cell Damage* – OCD – which evaluates the usefulness of input variables in a Multi-Layer Network and prunes the least useful. Variable selection is thus achieved while training the classifier, which ensures that the selected set of variables is adequate for the classifier. Variable selection is viewed here as a rather straightforward extension of weight pruning.

The second approach makes use of regularization terms under a Bayesian interpretation (Gaussian and Gaussian mixture priors) and allows to discard, during training, the least useful variables. Regularization and Bayesian training have been used recently in a variety of settings in NNs [32,17,37,23].

When implementing these methods, one has to decide on several options: when to start (pruning, or regularizing), how much (how many weights, or with how large a regularizing factor), which training algorithm to choose for ensuring the quality of the selection, and whether selection should be implemented as a 'non-convergent method' [14]. All of these decisions may indeed be important for the method applicability. In our tests, we discuss these issues.

We compare the performances of the two methods to the standard SAS *stepwise* and *stepdisc* procedures. This comparison is performed on two different tasks: a prediction problem for a synthetic time series and a classification problem from [6]. Both tasks are relatively simple, yet representative of the two major classes of problems addressed by NNs; they also are complex enough for our results to be significant.

The paper is organized as follows: Section 2 introduces notations and results from the literature; Section 3 the problem used to test our methods, Section 4 variable selection by OCD and Section 5 by regularization.

2. Variable selection

2.1. Definitions

Let be given a random variable pair $(X, Y) \in \mathbb{R}^P \times \mathbb{R}^Q$, with probability distribution P. Based on a sample $D_m = \{(x^1, y^1) \ldots (x^m, y^m)\}$, drawn from (X, Y), we train an NN to capture the relation between X and Y, i.e. to produce an estimator F, the NN approximation to the expectation $E[Y/X]$. We will deal in the following with prediction and classification tasks. The problem in variable selection is to extract the best set of features from an original input space, and possibly determine the optimal number p^* of features, for a given criterion: some components of the original input $X \in \mathbb{R}^P$ are eliminated to produce a vector with p features $x \in \mathbb{R}^p$, $p \leq P$. Training will be performed here according to a mean square error (MSE) criterion or an extended MSE criterion in the case of the Bayesian approach. We will use local criteria for ranking variables and perform selection during training. Efficiency of the selection will be measured through the empirical prediction error.

2.2. Statistical methods for variable selection / extraction

Feature selection has been extensively studied in the statistical or pattern recognition literature [15,26,27]. A feature selection technique typically requires the following ingredients:
- a feature evaluation criterion J to compare feature vectors $x \in \mathbb{R}^p$,
- a search procedure, to search the set of possible feature vectors x,
- a stop criterion, which could be a significance threshold in criterion J or the final feature space dimension.

Depending on the task (e.g. prediction or classification) and on the model (linear, logistic, ...), several evaluation criteria, based either on sound statistical grounds or heuristics, have been proposed for measuring the importance of a variable subset. For classification, classical criteria use probabilistic distances or entropy measures, often replaced in practice by simple interclass distance measures or even simple distances. For approximation or prediction, classical candidates are distance measures [19].

In general, these evaluation criteria are non monotonic, and exact comparison of feature subsets amounts to a combinatorial problem, which rapidly becomes computationally unfeasible, even for moderate input size. Due to these limitations, most algorithms are based upon heuristic performance measures and sub-optimal search. There are various procedures: in the *forward selection* method, vector x is progressively built up, by starting with the empty set and adding one or a few features at a time; the *backward* elimination technique works in the opposite direction: features are progressively eliminated from X one at a time.

Stopping the process is usually based on the evaluation criterion or on tests; one can also use a criterion for trading between complexity and accuracy such as AIC [1] or BIC [34].

In the following, our testbed will be the *stepwise* and *stepdisc* procedures implemented in SAS. Both alternate forward and backward selection, the evaluation criterion is the Fisher-Snedecor ratio, an F-test of the null hypothesis H_0 that all selected variables are significant (see Annex 1 for the description of these procedures).

In the NN literature, variable selection methods using entropy measures for feature evaluation have been proposed e.g. [2] they are based on the same ideas as the classical methods mentioned above and selection is performed prior to learning. The techniques we propose here are different in that they do selection after a first training step has been performed. They should be able to take into account the variable dependencies which have been estimated after training the system which will be used further on.

2.3. *Neural network methods for sensitivity analysis*

Various authors have proposed methods to *prune* useless connections so as to ensure correct generalization through the control of the network complexity [9,37]. We will focus here on the *Optimal Brain Damage* technique [22]: the *saliency* of a weight is defined as the cost variation resulting from the weight suppression.

Let N be a Neural Network picked from a family of multi-layer networks and D_m the learning set of size m. We use the Squared Error – SE – defined as:

$$C(W, D_m) = \tfrac{1}{2} \sum_{k=1}^{m} \| F_N(x^k) - y^k \|^2 \tag{2.1}$$

where $F_N(x^k)$ is the computed output for input x^k, y^k the desired output, and F_N the global transfer function realized by Network N.

The Mean Square Error – MSE – is then defined as:

$$MSE(W, D_m) = \frac{1}{2m} \sum_{k=1}^{m} \| F_N(x^k) - y^k \|^2 \tag{2.2}$$

Le Cun et al. [22] have shown that, if the Hessian can be approximated by a diagonal matrix, and the cost is locally quadratic, then the *saliency* of weight i is approximated, *locally around a minimum* W_0, by:

$$s_i = \tfrac{1}{2} H_{ii} W_i^2 = \frac{1}{2} \frac{\partial^2 C}{\partial W_i^2} W_i^2 \tag{2.3}$$

which can be computed by an additional backward pass to the usual Back-propagation algorithm (see Appendix 2). Exact methods could be used to compute the Hessian, instead of assuming diagonality [3,18].

The weight of smallest saliency is the weight whose pruning will least increase the cost. Notice that OBD explicitly requires that the network has reached convergence to a minimum W_0.

2.4. Neural networks and regularization

The regularization framework has provided another bunch of practical methods to handle the problem of poor generalization [31]. It considers estimation with too few and noisy data as an ill-posed problem, whereas constraints must be imposed on the final solution so as to make the problem well-posed. These constraints are usually implemented as follows. For an input x^k, desired output y^k and computed output $F(x^k)$, we train the network to minimize:

$$M(W) = \alpha\, C(W, D_m) + \beta E_W(W) \tag{2.4}$$

where $C(W, D_m)$ is as in (2.1) and the *regularization term* $E_W(W)$ embodies the constraints. Weight decay – WD – is the simplest regularization term used in the NN literature [32,37,8]:

$$E_W(W) = \sum_j W_j^2 \tag{2.5}$$

Such constraints may be given a Bayesian interpretation [13,5,23,24]. In this framework, one looks for weights W which maximize the a posteriori distribution, conditioned by the observed data D_m; which is equivalent, if one assumes an additive Gaussian noise model $N(0, \sigma^2)$ on data, to minimizing:

$$M(W) = \frac{1}{\sigma^2} C(W, D_m) - \ln P(W) \tag{2.6}$$

If prior $P(W)$ is taken to be gaussian $N(0, \sigma_W^2)$, then the log prior is, up to a constant, just weight decay (2.5) (where $|W|$ is the number of weights):

$$\ln P(W) = -|W| \ln \sigma_W \sqrt{2\pi} - \frac{1}{2\sigma_W^2} \sum_j W_j^2 \tag{2.7}$$

Other assumptions on the weights prior can be made [7,30]. This Bayesian framework can be used for pruning [25,35]. Regularization hyper-parameters α and β have to be determined: they have an interpretation in terms of the data noise and weight variances. The full Bayesian approach can in theory provide for their automatic determination. However, for large dimensions, this approach is not efficient, so that usually, they are set by using some form of cross-validation. In the following, we use a regularized cost function to perform feature selection, with two different weights priors: Gaussian – WD – and Gaussian mixture. All hyper-parameters are determined by using a validation set.

2.5. Summary

Conventional techniques for variable selection are hindered by the non-monotonicity problem: heuristics guide both selection and stopping criteria. They usually build-up features sets by including – or excluding – one – or a few – feature(s) at a time: because of non-monotonicity, this can lead to sub-optimal subsets. Various parameters have to be set a priori, e.g. the number of features added – or removed – at each step, the final features space size, significance level for selecting features...Final results can widely vary depending on the choices made. After selection has been made, classification or prediction may rely on totally different techniques. On the other hand, the heuristic approach we propose here allows to carry on simultaneously variable selection and parameter estimation for the desired task: variables are selected globally to best fit the final estimator. However, NN training requires the setting of various parameters and hyperparameters in the case of regularization: no exact theory so far can provide for the exact setting of these parameters. One can resort to different heuristics to set them: trial-and-error, cross-validation, estimation or learning. In the following, we will chiefly use cross-validation. Whatever the method selected, choosing the parameters usually requires a large amount of additional computations.

3. Experiments setting

In this section, we describe the version of Gradient Back Propagation – GBP – and the test problem we used. It is important here to describe the precise version of GBP which has been implemented since it may influence the quality of convergence and thus the feature selection process. For example, performances on our time series are very sensitive to GBP parameters.

3.1. Training procedure

We use three independent sets of respective sizes m_l, m_v and m_t: $D^l_{m_l}$ for learning, $D^v_{m_v}$ for validation (i.e. set the parameters) and $D^t_{m_t}$ for test. Our training procedure is an on-line GBP implemented as follows: the gradient step ε is started from an initial value ε_0 and decreased as soon as the relative average cost variation becomes too small. It has been chosen here for its simplicity.

```
GBP • run N steps of on-line GBP with $\varepsilon = \varepsilon_0$.
    • while $\varepsilon \geq \varepsilon_1$ do
          run on-line GBP on $D^l_m$ with $\varepsilon$
          if $r(W, D^v_m, t, T_1) \leq \theta_1$
              then $\varepsilon(t+1) = \alpha.\varepsilon(t)$ else $\varepsilon(t+1) = \varepsilon(t)$
    endwhile
```

where

$$r(W, D, t, T) = 1 - \frac{[C(W, D)]^T_t}{[C(W, D)]^T_{t-1}} \tag{3.1}$$

$$[C(W, D)]^T_t = \frac{1}{T} \sum_{\tau=0}^{T-1} C(W(t-\tau), D)$$

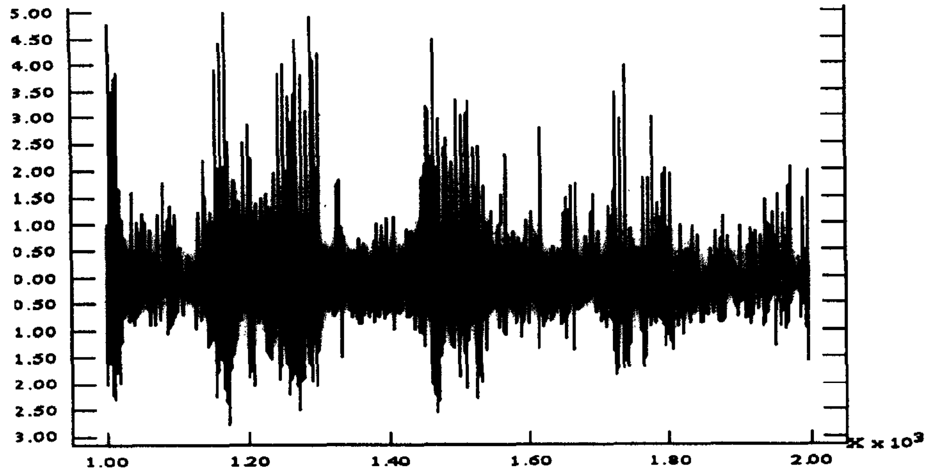

Fig. 1. Time series z_t from validation set D^v_m.

$r(W, D, t, T)$ is the relative average cost variation between epochs $(t-T$ to $t-1)$ and $(t-T+1$ to $t)$. $C(W, D)$ is the error defined in (2.1).

GBP thus depends on various parameters: ε_0, ε_1 and θ_1 being given, N, T_1 and α are set by cross-validation on the validation set, using the MSE criterion. The methods described below were found sensitive to the quality of local minima and thus to the above parameters. Note that such parameters do exist for any gradient-like algorithm. Weight initialization also influences the algorithm behavior, but this has not been considered here.

3.2. *Time series*

We have used a non linear synthetic time series y_t, defined as follows:

$$x_t = 0.3x_{t-6} - 0.6x_{t-4} + 0.5x_{t-1} + u_t$$

$$y_t = x_t + 0.3x^2_{t-6} - 0.2x^2_{t-4} \qquad (3.2)$$

where u_t is a white noise.

Three independent samples of size $m = 1000$ each are drawn from series y_t: D^l_m, D^v_m and D^t_m. Each sample set is normalized as follows:
- the average $\bar{y}$ and variance s^2 of set D^l_m are computed.
- y_t is then changed to $z_t = (y_t - \bar{y})/(s)$, for all 3 sets D^l_m, D^v_m and D^t_m.

Fig. 1 shows set D^v_m.

Best performances on this series may be obtained with a linear network whose inputs are the predictors of (3.2). We have used a linear network 12-1 (with all 10 input variables x_{t-10} to x_{t-1} plus variables x^2_{t-6}, and x^2_{t-4}): variable selection should pick up the 'true' inputs x_{t-6}, x_{t-4}, x_{t-1}, x^2_{t-6}, x^2_{t-4} since y_t is linear in these variables. The final MSE should be approximately the variance of u_t, i.e.

T_1	3			5			7		
α\Set	D_m^l	D_m^v	D_m^t	D_m^l	D_m^v	D_m^t	D_m^l	D_m^v	D_m^t
3/4	0.04376	0.05051	0.04676	0.04359	**0.05049**	0.04672	0.04345	0.05055	0.04670
5/6	0.04341	0.05056	0.04670	0.04331	0.05065	0.04673	0.04327	0.05072	0.04677
7/8	0.04352	0.05095	0.04655	0.04334	0.05085	0.04664	0.04309	0.05056	0.04628
net 12-1	0.04461	0.04629	0.04209						

Fig. 2. Final MSE for time series, net 10-7-1 (3 top lines) and 12-1 (bottom).

0.04072 (after normalization of y_t). This network represents the best possible performances and will be used here for comparison.

Because in real situations we do not know the underlying structure of the series, we used for implementing our selection methods a network 10-7-1 (with all 10 variables x_{x-10} to x_{t-1} as inputs, 7 non linear hidden units and 1 linear output). Now, selection should pick up input variables $(x_{t-6}, x_{t-4}, x_{t-1})$.

Results are shown in Fig. 2 for net 10-7-1 and net 12-1. We use the MSE on D_m^v to choose the best parameter values for net 10-7-1 (for the sake of simplicity, only results for choosing T_1 and α are given; results allowing to choose N are of the same sort). Net 12-1 is very easy to train, and its performances are extremely robust with respect to parameters choice.

Cross-validation gave (net 10-7-1) $T_1 = 5$, $N = 30$, and $\alpha = 3/4$, for $\varepsilon_0 = 0.1$, $\varepsilon_1 = 0.0001$, $\theta_1 = 0.0005$. Values $\varepsilon_0 = 0.1$, $\varepsilon_1 = 0.000005$, $\theta_1 = 0.000025$ (to reach almost perfect convergence), $\alpha = 0.875$ were used for net 12-1, without doing any cross-validation there (see Fig. 2).

Final MSEs at the end of GBP with net 12-1 were of order 45.10^{-3} (which is slightly larger than $var(u_t)$, as expected). Errors bars on the final MSE should be computed. For more clarity in the figures, we have not given them in the text: the interested reader can refer to Appendix 3 where our technique for computing error-bars is described, together with a few results. The time series y_t has some pseudo-periodic behavior, which results in D_m^v being very oscillating, and D_m^t much smoother. This explains why performances on test set D_m^t were often found better than on validation set D_m^v.

3.3. Waveforms

We have extended a problem introduced in [6]: 3 vectors or *waveforms* in 21 dimensions, H^i, $i = 1,\ldots,3$, are given. Patterns in each class are defined in $\mathbb{R}^{40}$ as random convex combinations of 2 of these vectors (waves (1, 2), (1, 3), (2, 3) respectively for class 1, 2 and 3, see Fig. 3) plus 19 additional noisy components [4]. The problem is then to classify these patterns into one of the 3 classes. More precisely, patterns are generated according to:

$$x_i = \frac{uH_i^m + (1-u)H_i^n}{5} + \varepsilon_i \qquad 0 \le i \le 20$$

$$x_i = \varepsilon_i \qquad 21 \le i \le 40$$

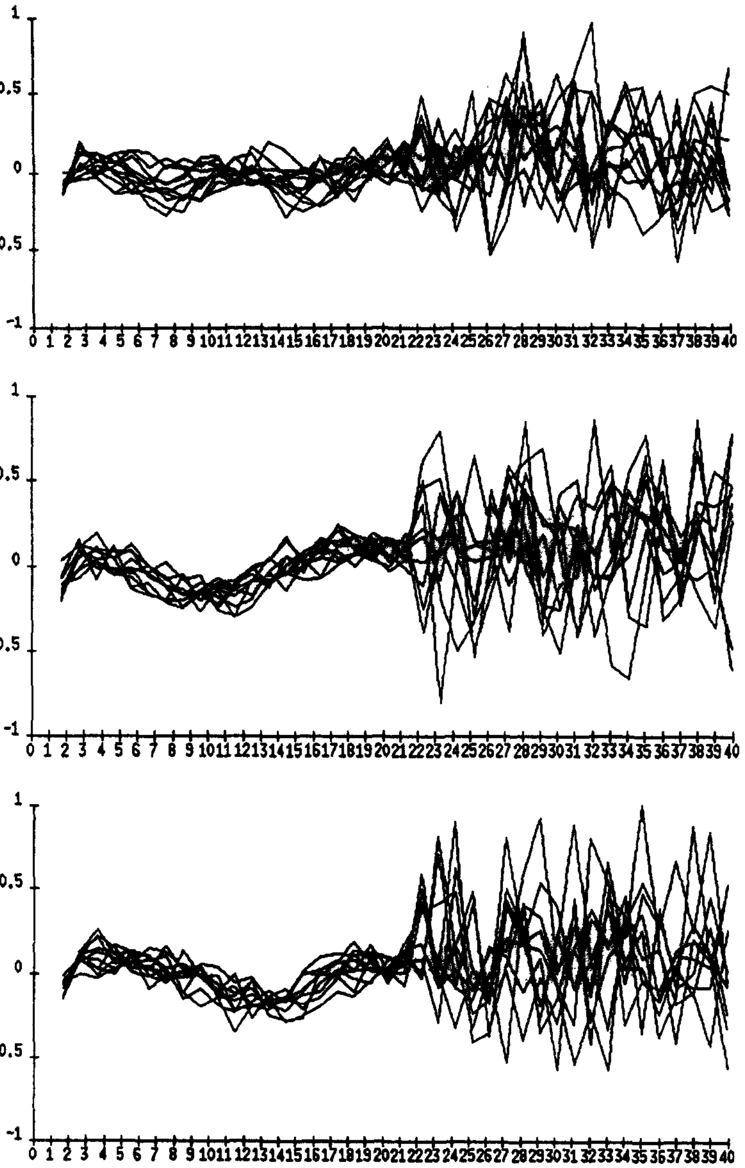

Fig. 3. Waveforms, each graph plots several patterns from class 1, 2 and 3 (top to bottom). The first 21 components contain the class information, the last 19 are noisy components.

Set	D^l_m	D^v_m	D^t_m
40-10-3	95.67	80.40	80.36
40-3	93.66	81.40	80.18

Fig. 4. Final MSE for waveforms, net 40-10-3 (top) and 40-3 (bottom).

where x_i denotes the ith component of a pattern x, u is a uniform random variable in [0, 1], ε_i is generated according to a normal distribution $N(0, 1)$, m and n identify the two waves used in this combination, i.e. the class of pattern x. D_m^l has 300 elements, D_m^v 1000 and D_m^t 5000.

Fig. 4 gives the final performances at the end of GBP with net 40-10-3 and net 40-3. These were obtained for the following values set after cross validation: $T_1 = 10$, $\alpha = 0.85$, $\varepsilon_0 = 0.2$, $\varepsilon_1 = 0.0001$, $\theta_1 = 0.0001$ (net 40-10-3) and $T_1 = 30$, and $\alpha = 0.95$, for $\varepsilon_0 = 0.2$, $\varepsilon_1 = 0.0001$, $\theta_1 = 0.0001$ (net 40-3).

4. Optimal cell damage

4.1. The algorithm

We now describe our variable selection procedure which is based on an extension of OBD. The cost variation can be approximated to order 2, around a local minimum W_0 by a quadratic form [22] (Appendix 2). Discarding a variable x_i can be implemented by setting to 0 all weights W_{ji} in Fan_Out(i), the Fan-Out of input neuron i. Thus the resulting cost is just the sum of the costs associated to the suppression of the various weights. Similar methods have been proposed for discarding hidden units [9].

The *saliency* of variable i is thus defined as:

$$\zeta_i = \sum_{j \in Fan\_Out(i)} s_j \tag{4.1}$$

where s_j is the saliency of weight W_{ji} (2.3).

Our algorithm for OCD has been run as follows (after normalization of data as indicated in Section 3): if the relative average variation of cost in two successive periods is too small, then prune a fraction q of inputs.

```
OCD   if $r(W, D_m^v, t-1, T_1) \le \theta_2$ and $r(W, D_m^v, t, T_1) \le \theta_2$        (4.2)
   then    • compute saliencies $\zeta_i$ at time $t$ and order them: $\zeta_{i_1} \ge \zeta_{i_2} \ge \ldots \ge \zeta di_p$
           • choose $p$ such that: $\sum_{l=1}^{p} \zeta_{il} / \sum_{l=1}^{P} \zeta_{il} \ge q$ and eliminate variables $i_{p+1}, \ldots, i_P$
   continue OCD
else continue GBP
```

Pruning starts early if θ_2 is large, late otherwise. Both parameters θ_2 and q are determined by cross-validation.

To assess the quality of the selection, we should think of the practical use of variable selection: typically one uses a relatively small data set, to decide upon the best subset of variables; then one gathers a larger data set, where only the selected subset of variables is measured, and train a network on this data set. If the selected subset is good, the new network (trained on smaller dimension inputs) should be as good – or even better – than the original network trained on the full dimension inputs.

q	0.99			0.9925			0.995		
θ_2	D_m^l	D_m^v	D_m^t	D_m^l	D_m^v	D_m^t	D_m^l	D_m^v	D_m^t
0.1	0.04489	**0.04793**	0.04454	0.04453	0.04908	0.04530	0.04450	0.04960	0.04569
selected	4, 6, 1			4, 6, 1, 2			4, 6, 1, 2, 10		
0.05	0.04489	**0.04793**	0.04454	0.04453	0.04908	0.04530	0.04450	0.04960	0.04569
selected	4, 6, 1			4, 6, 1, 2			4, 6, 1, 2, 10		
0.01	0.06176	0.06673	0.06052	0.04456	0.04911	0.04529	0.04475	0.04977	0.04587
selected	4, 6, 1			4, 6, 1, 2			4, 6, 1, 2		

Fig. 5. Final MSE and selected variables for time series after OCD.

selected set	4, 6, 1	4, 6, 1, 10	4, 6, 1, 2, 7		
Net	3 - 7 - 1	4 - 7 - 1	5 - 7 - 1	10 - 7 - 1	12 - 1
D_m^l	0.04553	0.04495	0.04465	0.04359	0.04461
D_m^v	0.04851	0.04790	0.04832	0.05049	0.04629
D_m^t	0.04533	0.04488	0.04451	0.04672	0.04209

Fig. 6. Performances of networks retrained on the selected sets of variables for Time series.

4.2. *Time series*

For time series we ran our *OCD* algorithm with $T_1 = 5$ and $\alpha = 3/4$ (as determined in Section 3.2). Results are shown in Fig. 5: for the optimal values $q^* = 0.99$ and $\theta_2 = 0.05$ or 0.1, our variable selection procedure has selected the appropriate variables x_{t-6}, x_{t-4}, x_{t-1}: (variables 1, 4 and 6 in Fig. 5).

	q	0.975			0.985			0.995		
	θ_2	D_m^l	D_m^v	D_m^t	D_m^l	D_m^v	D_m^t	D_m^l	D_m^v	D_m^t
40-10-3	**0.01**	90.00	73.70	82.34	92.00	**84.00**	83.30	94.33	81.80	81.96
	nb sel. var	10			14			21		
	0.001	90.33	80.00	83.46	93.67	83.10	83.28	95.67	81.40	81.78
	nb sel. var	11			15			27		
	0.0001	93.00	82.20	82.54	93.33	82.20	81.80	94.67	80.60	80.94
	nb sel. var	17			19			25		
40-3	**0.01**	91.33	79.70	84.08	92.67	**84.60**	84.36	94.67	83.90	83.74
	nb sel. var	11			14			16		
	0.001	90.33	80.30	84.02	92.67	84.10	84.54	94.00	82.60	83.48
	nb sel. var	11			14			17		
	0.0001	91.33	80.00	84.10	92.33	84.00	84.30	93.33	82.50	83.16
	nb sel. var	11			14			18		

Fig. 7. Final MSE and selected variable number for Waveforms after OCD.

Set	method	nb.sel.var	1:var. selected, 0: eliminated
0	true signal -A priori information	21	111111111111111111111 0000000000000000000
1	OCD - 40-3	14	000111101111111111000 0000000000000000000
2	OCD - 40-10-3	14	000111101111111110100 0000000000000000000
3	WD - 40-3	17	001111101111111111110 0000000000000000000
4	WD - 40-10-3	19	101111111111111111110 0000000000000000000
5	Reg.CV - 40-3	15	000111101111111111100 0000000000000000000
6	Reg.CV - 40-10-3	17	001111111111111111100 0000000000000000000
7	Reg.Est - 40-3	15	000111101111111111100 0000000000000000000
8	SAS (Stepdisc)	14	000110100111100011100 0000000100000000111

Fig. 8. Selected variables for the different selection methods on Waveforms. *M-X*: method *M* on network $X(X = 40\text{-}10\text{-}3$ or 40-3). OCD: OCD (Section 4.3); WD: simple Weight Decay (Section 5.2.2); Reg.CV: regularization with Gaussian mixture (Section 5.3.2); Reg.Est: regularization with hyperparameters estimated during training (Section 5.4.2).

As can be seen in Fig. 5, pruning variables also has the side effect of decreasing the final MSE.

The stepwise procedure in SAS also selected the set of variables 1, 4, 6 (see Appendix 2). From which it follows that OCD has similar performances as SAS stepwise (in terms of the selected subset): in fact, OCD is less efficient, since it requires quite a large overhead in computing time.

Fig. 6 shows the benefit of variable selection: network 3-7-1 retrained with only the selected variables achieves a lower final MSE than the original net 10-7-1.

	"set"-3			"set"-10-3		
Set	D^l_m	D^v_m	D^l_m	D^l_m	D^v_m	D^t_m
0	92.00	85.90	85.50	92.67	84.80	85.76
1	91.67	85.50	84.90	91.33	85.00	85.10
2	91.67	85.50	84.96	93.00	85.00	84.90
3	93.00	84.60	85.28	93.33	85.10	85.64
4	92.67	85.90	85.46	92.67	84.90	85.30
5	92.33	84.80	85.02	92.67	85.00	85.00
6	92.33	85.80	85.34	93.00	84.50	85.06
7	92.33	84.80	85.02	92.67	85.00	85.00
9	90.33	80.40	79.56	90.33	79.90	79.84
9:All 40 variables	93.66	81.40	80.18	95.67	80.40	80.36

Fig. 9. Performances of networks retrained on the selected sets of variables for Waveforms. The left column x designates the set of variables selected by the corresponding method (Fig. 8). Columns 'set' – 3 (resp. 'set'-10-3) give the performances of network y-3 (resp. y-10-3), where y is the number of variables selected in set x. Networks y-3 and y-10-3 are retrained on set x.

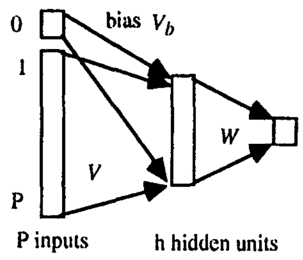

Fig. 10. MLP used to produce our predictor.

4.3. *Waveforms*

For the waveforms, we ran our OCD algorithm with values of T_1 and α as determined in Section 3.3, both for network 40-10-3 and 40-3. Results are shown in Fig. 7: for the optimal values $q^* = 0.985$ and $\theta_2 = 0.01$, our variable selection procedure selected 14 variables among the 21 first - the signal - (the same for both networks, except for one variable) and eliminated all the noise (the 19 last variables): see Fig. 8. Note that SAS did not succeed in eliminating all the noise. Error bars are computed as indicated in Annex 3.

As can be seen in Fig. 9, variable selection increases performances by more than 4% compared to SAS selection.

5. Regularization

5.1. *Introduction*

We will restrict our analysis here to the case of a multilayer perceptron (MLP) with only one hidden layer, and full connections. Extension to more complex architectures is straightforward although its application may be more difficult. Let us suppose that our MLP (Fig. 10) has P inputs, h hidden units and p output units, and denote V, resp. W, the weights from input to hidden, resp. hidden to output layer and V_b the bias vector to hidden units (W includes the bias to the outputs). V_i is thus the weight vector exiting from unit i discarding feature i is equivalent to setting V_i to 0.

With these notations, Eq. (2.6) becomes:

$$M(V_b, V, W) = \frac{1}{2\sigma^2} \sum_{k=1}^{m} \| F(x^k) - y^k \|^2 - \ln P(V_b, V, W) \tag{5.1}$$

We will use two different priors $P(V_b, V, W)$: a Gaussian and a Gaussian mixture.

5.2. Simple weight decay

5.2.1. The algorithm

To select input variables, we will take a Gaussian for $P(V)$, i.e.:

$$\ln P(V_b, V, W) \propto -\frac{1}{2\sigma_V^2}\sum_{i=1}^{P}\sum_{j=1}^{h} V_{ji}^2 \tag{5.2}$$

This is almost identical to weight-decay, except that we only constrain the input-to-hidden weights, and not the other weights. This reflects our goal of pruning those weights, while we do not have, a priori, any constraint on the remaining weights.

Our variable selection technique run as follows: first use a `Regularization` algorithm, then a `Variable selection` algorithm. In the `Regularization` algorithm, we first start by a few steps of the usual Square Error cost (to start fitting the data), then move on to the regularized cost function M of (5.1):

```
Regularization                                                    (5.3)
    • run N steps of on-line GBP with $\varepsilon = \varepsilon_0$ and cost function $C$.
    • from now on use cost function $M$.
    • while $\varepsilon \geq \varepsilon_1$ do
          run on-line GBP with $\varepsilon$
          if $r(W, D_m^v, t, T_1) \leq -\theta_1$
              then $\varepsilon(t+1) = \alpha.\varepsilon(t)$ else $\varepsilon(t+1) = \varepsilon(t)$
    endwhile
```

where r is defined as before by (3.1), i.e. using function C.

This algorithm is executed, with the values of N, T_1 and α as set by cross-validation after GBP (see Section 3.2.), for the given values of ε_0, ε_1 and θ_1. The first N steps are exactly as before; from N on, regularization starts.

Let $\lambda = \sigma^2/\sigma_V^2$, this parameter must be estimated from the data. Usually we have no knowledge about σ, so we used cross validation to set parameter λ^*, by comparing the MSEs on D_m^v and selecting the parameter value corresponding to the minimum error on D_m^v. From now on, cost function M will use that value of λ^*.

Let us then define, for every input variable i, its weight variance var_i. `Variable selection` is run as follows (5.4): p variables are selected so as to retain a proportion q of the total variances var_i. GBP and `Variable selection` are thus alternated until the stopping criterion for GBP is met. q is set by cross validation:

```
Variable selection                                                (5.4)
    • order variances: $var_{i_1} \geq var_{i_2} \geq \ldots \geq var_{i_P}$
    • choose smallest p such that: $\Sigma_{k=1}^{p} var_{i_k} \geq q\Sigma_{k=1}^{P} var_{i_k}$
```

λ / Set	0	**0.0001**	0.005	0.001	0.1	q	0.85	0.89	**0.92**	0.94
D_m^l	0.04359	0.04360	0.04455	0.04556	0.16522		0.07008	0.04765	0.04634	0.04606
D_m^v	0.05050	**0.05029**	0.05105	0.05265	0.19367		0.07405	0.05133	**0.05014**	0.05035
D_m^t	0.04672	0.04669	0.04743	0.04933	0.19979		0.06962	0.04766	0.04698	0.04729
selected							4, 6	4, 6, 1	4, 6, 1, 10	4, 6, 1, 10, 8
" by SAS						4, 6, 1				

Fig. 11. Weight Decay on Time series: determination of λ^* after Regularization and q^* after Variable selection. The subset of variables selected in each case is shown in the bottom lines.

- eliminate variables i_{p+1} to i_P
- run N′ steps of on-line GBP with $\varepsilon = \varepsilon_1$ and cost function M.

5.2.2. *Time series*

Results are shown in Fig. 11.

Results (Fig. 6) show that our WD-based selection indeed worked well. If we use as inputs the variables selected by WD (from results shown in Fig. 11, we have selected the 4 variables x_{t-6}, x_{t-4}, x_{t-1} and x_{t-10}), and retrain a network 4-7-1 with only these inputs, we obtain a final MSE lower than both the initial 10-7-1 network and the network 3-7-1 retrained on the variables selected by SAS or OCD (for an analysis of the significance of these results, see Appendix 3 where error bars are provided).

It might seem strange that the best choice would be to select, beside the obvious variables x_{t-6}, x_{t-4}, x_{t-1}, variable x_{t-10} also, which does not appear in the expression of (3.2) for y_t. However, if one writes expression (3.2) for variable x_{t-4}, variable x_{t-10} will indeed appear: in fact x_{t-10} is implicitly present in variable y_t, and our selection process has been able to take advantage of this implicit correlation.

Set /λ	0.00005	0.0001	**0.0005**	0.001	0.1	q	0.89	0.92	0.96	0.98
D_m^l	94.67	94.67	**95.22**	95.22	93.56		92.00	**92.67**	93.67	96.67
D_m^v	80.23	80.20	**80.47**	80.00	80.33		83.20	**83.40**	82.20	81.30
D_m^t	80.22	80.39	**80.51**	80.41	80.18		84.38	**85.00**	82.92	82.56
NbSelVar							16	19	27	32
D_m^l	93.00	93.30	**93.00**	93.00	83.67		86.67	92.33	**90.33**	93.00
D_m^v	81.00	81.00	**81.60**	81.00	72.00		82.00	83.60	**84.60**	83.50
D_m^t	80.32	80.16	**80.28**	80.24	72.26		79.22	84.34	**84.08**	83.28
NbSelVar							13	14	17	23

Fig. 12. Weight Decay on Waveforms: determination of λ^* after Regularization and q^* after Variable selection. The upper part of the table corresponds to net 40-3, the lower part to 40-10-3. The number of variables selected in each case is shown in the bottom lines (NbSelVar).

These results also shed light on the validity of our a-priori assumption on the weights: a Gaussian prior is certainly reasonable, since it allows us to improve on our predictor's accuracy. The question is now to see whether we could find a better prior: we will turn, in Section 5.3, to a Gaussian mixture prior.

5.2.3. Waveforms

Results in Fig. 12 show the performances of our WD-based selection. Note that the Weight Decay technique selected more variables than OCD or SAS, although always in the 'signal' part: it made use of these for increased performances on test set. The Gaussian prior assumption is in this sense validated.

5.3. Gaussian mixture prior

5.3.1. The algorithm

Assume a priori that weights V and W are independent (which is not true any more as soon as learning starts!) and that W, resp. V_b is Gaussian $N(0, \sigma_W^2)$, resp. $N(0, \sigma_b^2)$. V_i's are supposed to be independently distributed with a Gaussian mixture of two components: one – in proportion π - $N(0, \sigma_l^2)$ has a large variance σ_l^2 and the other – in proportion $(1-\pi) - N(0, \sigma_s^2)$ a small one σ_s^2. Under these assumptions, we have:

$$\ln P(V_b, V, W) \alpha - \frac{1}{2\sigma_W^2}\sum_{s=1}^{Q}\sum_{j=0}^{h} W_{sj}^2 - \frac{1}{2\sigma_b^2}\sum_{j=1}^{h} V_{bj}^2 + \sum_{i=1}^{P}\ln\left[\frac{\pi}{\sigma_s^h} e^{-\frac{1}{2\sigma_s^2}\sum_{j=1}^{h} V_{ji}^2} + \frac{1-\pi}{\sigma_l^h} e^{-\frac{1}{2\sigma_l^2}\sum_{j=1}^{h} V_{ji}^2}\right] \tag{5.5}$$

M(V_b, V, W) is minimized, through an on-line GBP (see Section 3). We have to determine the following parameters: σ_W^2, σ_l^2, σ_s^2, σ_b^2, π, q and σ^2. There do not exist, at the present moment, exact techniques to compute optimal values of the hyperparameters: the Bayesian optimization approach [24], for example, relies on assumptions and is very computation expensive.

We have tested various methods to determine the hyperparameters: fixing some and determining the others by cross-validation or fixing some and estimating others. We describe some results of these different techniques in the following paragraphs.

π / Set	0	0.4	0.6	0.7	0.8	q	0.85	0.98	0.9993	0.9995
D_m^l	0.04503	0.04530	0.04531	0.04531	0.04532		0.06798	0.04643	0.04585	0.04566
D_m^v	0.05108	**0.04857**	0.04860	0.04861	0.04861		0.07103	0.04899	**0.04820**	0.04838
D_m^t	0.04768	0.04575	0.04578	0.04578	0.04579		0.06759	0.04673	0.04556	0.04566
selected							4, 6	4, 6, 1	4, 6, 1, 2, 7	4, 6, 1, 2, 7, 8

Fig. 13. Time series: determination of π^* with `Regularization` (with Gaussian mixture) and of q^* with `Variable selection`.

5.3.2. Time series

In this paragraph, all hyperparameters are held constant ($\sigma_W^2 = \sigma_l^2 = \sigma_b^2 = 1$, $\sigma_s^2 = 0.005$), while weights V and W are updated; π, σ^2 (in 5.1) are set by cross-validation during algorithm Regularization: only results with the best σ^2 are given. Then, as for weight decay, the Variable selection algorithm is run and q is set by cross-validation. Results are shown in Fig. 13.

Comparison of performances of the net 5-7-1 retrained with the selected variable subset (Fig. 6) shows that, finally, it is the set of variables selected by this Gaussian mixture regularization which gives the best performances on test set. The analysis of error bars (see Appendix 3) shows that net 5-7-1 has an MSE significantly smaller than that of net 3-7-1: thus our variable selection approach with regularization and a Gaussian mixture prior improves upon both OCD and stepwise.

Notice that our technique has learned to use both 'explicit' variables 4, 6 and 1 (x_{t-4}, x_{t-6}, x_{t-1}) and 'implicit' variables 2, 7 (x_{t-2}, x_{t-7}) coming from y_{t-1}.

5.3.3. Waveforms

For this problem, we have tested the effectiveness of cross-validation for setting the mixture parameters. Hyperparameters π and σ^2 are thus fixed; σ_l^2, σ_s^2 (and σ_W^2 for net 40-10-3) are determined by cross-validation with Regularization and q with Variable selection. In this case, best performances were obtained for very high values af the variances, which is equivalent to zeroing the mixture a priori hypothesis (third term of Eq. (5.5) right member). Cross-validation thus converged to a trivial solution where no regularization is used for the input to hidden weights. One reason may be that in this case, variable selection is relatively easy and simple procedures like Variable selection perform well enough. Note however that this is not the case with the default stepdisc SAS procedure which has been used here.

5.4. Hyperparameters estimation

Normally all parameters should be set appropriately. Various methods exist: learning, estimation, cross-validation or exact computation, e.g. through bayesian theory [25]. Cross-validation, which we have used in this paper, is very computation

Set \ Method	Cross-validation	Estimation
D_m^l	0.04585	0.04485
D_m^v	0.04820	0.04750
D_m^t	0.04556	0.04378

Fig. 14. Time series, final MSE reached after Regularization, when parameters are set by cross-validation (left) or estimated (right).

Input unit	var_i	Input unit	var_i
10	0.000001	5	0.000001
9	0.000001	4	0.343770
8	0.000001	3	0.000001
7	0.000001	2	0.000002
6	0.263500	1	0.018315

Fig. 15. Time series, weight variances of input units after training with Regularization.

intensive. One can thus try to estimate some of these parameters. We have estimated hyper-parameters for the two problems. We present below an example of such an approach for time series. This approach worked well also for the classification problem but gave performances similar to the other methods (see Figs. 8 and 9).

For the time series, we set $\sigma_W^2 = \sigma_l^2 = 1$ fixed a priori. We then train N steps with cost function C. We then use as an estimate for σ^2 the MSE on D_m^v, a new estimate being computed after each epoch. σ_s^2 is progressively decreased through learning. Results (Fig. 14) are very good in terms of final MSE, as compared to the final MSE obtained after Regularization for our previous setting of parameters.

In addition, if one look at the weights variances var_i of the 10 input units, one can easily select the most significant variables: since there is at least a 4 order of magnitude difference (see Fig. 15).

However, we see that the algorithm has just picked the 'obvious' solution, and not the better ones found out by the previous methods. Additional work is certainly needed to explore more systematically the various methods for hyper-parameters setting.

6. Conclusion

We have presented here a technique based on regularization for variable selection, with two different regularizing terms, coming from different hypothesis on the weights priors (Gaussian and mixture of Gaussians). The regularization with gaussian mixture gave the best final performances on test set for a synthetic time series example, as compared to the WD-based method, a conventional statistic method (stepwise in SAS) or a pruning method (OCD).

For the classification example, the three NN based methods gave similar performances, and are significatively better than standard classical techniques.

The methods we propose still suffer from various problems:

- There are many parameters which have to be set by cross validation. This is far too computation intensive in practical problems. In particular, our version of GBP should be replaced in the case of approximation or identification problems such as the time series in this paper, by a parameter-free technique, e.g. conjugate gradients [28].
- The results are very sensitive to the a priori choices of parameters (e.g. σ^2, σ_W^2, σ_l^2, σ_s^2).

– Input correlation is not taken into account, which means that even linearly dependent parameters may be selected.

Appendix 1. Stepwise procedure

Let $k \leq N$ be the size of feature space at some given time: we want to know whether to retain all k variables or discard $k-q$ of them. Let F be defined as:

$$F = \frac{m-k}{k-q} \frac{\| \underline{\hat{Y}}_k - \underline{\hat{Y}}_q \|^2}{\| \underline{Y} - \underline{\hat{Y}}_k \|^2}. \tag{A.1}$$

where $\underline{Y}$ is the sample output vector $(Y^1, \ldots, Y^m)^t$, $\underline{\hat{Y}}_i$, $i = q, k$, is the estimation produced when using i variables:

$$\underline{\hat{Y}}_i = \left(F_{\alpha_i}(X^1), \ldots, F_{\alpha_i}(X^m)\right)^t \tag{A.2}$$

Let θ_α be given by:

$$P(F > \theta_\alpha) = \alpha \tag{A.3}$$

where F is a Fisher-Snedecor variable with $(k-q,\ m-k)$ degrees of freedom and α is some confidence level (e.g. $\alpha = 0.95$). Then, if the measured ratio F (A.1) is larger than θ_α, all k variables are kept, otherwise q variables only are: this is a F-test of the null hypothesis H_0 that the $k-q$ variables are significant.

$$\begin{array}{ll} F > \theta_\alpha & \text{accept } H_0\text{: use all } k \text{ variables} \\ F < \theta_\alpha & \text{reject } H_0\text{: use } q \text{ variables, reject the other } k-q \text{ variables} \end{array} \tag{A.4}$$

For a given size n of the feature space, selecting the best subset x of n features out of the N possible $(X_1, \ldots, X_N)$ is a combinatorial search problem (there are (N!)/(n!(N – n)!) such subsets). So exhaustive search is not feasible. If the feature selection criterion J is monotonous, that is:

$$\aleph_1 \subset \aleph_2 \subset \cdots \subset \aleph_N \Rightarrow J(\aleph_1) \leq J(\aleph_2) \leq \ldots \leq J(\aleph_N) \tag{A.5}$$

where $\aleph_k$ contains k variables X_i, then a simple branch-and-bound algorithm [19] allows to restrict the search and efficiently come up with an optimal feature vector. However, most selection criteria are not monotonous, and thus only suboptimal procedures are known.

For example, the sequential forward method progressively builds up a vector x, from the empty set, by adding one feature at a time, selected to maximise criterion J. More precisely, if $\aleph_k$ is the feature vector at instant k, a feature x_{k+1} is selected in $X - \aleph_k$ so that:

$$J(\aleph_k \cup \{x_{k+1}\}) = \max_{x_i \in X - \aleph_k} J(\aleph_k \cup \{x_i\}) \tag{A.6}$$

The sequential backward technique is similar, in opposite direction: starting from the complete vector X, features are eliminated one at a time so as to maximise the criterion:

$$J(\aleph_{n-k} - \{x_{k+1}\}) = \max_{x_i \in \aleph_{N-k}} J(\aleph_{N-k} - \{x_i\}) \tag{A.7}$$

where $\aleph_{N-k}$ is the feature vector at iteration k.

Variable i	F-value	P > F
6	108.374	0.0001
1	6.54	0.0110
4	6.46	0.0115

Fig. 16. Stepwise procedure. $P > F$ is the probability that the F stat. is superior to the F value.

The F-factor can be used as a criterion J; Eqs. (A.6) and (A.7) then become:

$$F_i = (m-k-1)\frac{\|\hat{\underline{Y}}^i_{k+1} - \hat{\underline{Y}}_k\|^2}{\|\underline{Y} - \hat{\underline{Y}}^i_{k+1}\|^2}, \quad x_{k+1} = \operatorname{Arg}\max_{x_i \in X - \aleph_k} F_i \tag{A.8}$$

where $\aleph_k \subset \aleph^i_{k+1} = \aleph_k \cup \{x_i\}$ and $\hat{\underline{Y}}_k$, resp. $\hat{\underline{Y}}^i_{k+1}$, is the estimator produced using variables in $\aleph_k$, resp. $\aleph^i_{k+1}$.

$$F_i = (m-N+k)\frac{\|\hat{\underline{Y}}_{N-k} - \hat{\underline{Y}}_{N-k-1}\|^2}{\|\underline{Y} - \hat{\underline{Y}}_{N-k}\|^2}, \quad x_{k+1} = \operatorname{Arg}\max_{x_i \in \aleph_{N-k}} F_i \tag{A.9}$$

where $\aleph_{N-k} \supset \aleph^i_{N-k-1} = \aleph_{N-k} - \{x_i\}$ and $\hat{\underline{Y}}_{N-k}$, resp. $\hat{\underline{Y}}^i_{N-k-1}$, is the estimator produced using variables in $\aleph_{N-k}$, resp. $\aleph^i_{N-k-1}$.

The `stepwise` starts from an empty variable subset, and then alternates the `sequential forward` and `backward` procedures. We run these procedures with a confidence level of 0.85, and entering/eliminating one variable at a time.

For example, the `stepwise` procedure enters the variables as shown in Fig. 16, and then no other variable met the 0.85 confidence level for entering or exiting the model.

Stepdisc procedure

The Stepdisc procedure implemented in SAS for selecting variables for discrimination tasks relies on the hypothesis that the classes are multi-normal with equal covariance matrices. The criterion for variable selection is the minimization of Wilks' lambda λ_W: for q variables,

$$\lambda_W = \frac{|W_q|}{|T_q|}$$

where W_q, T_q are respectively the within and total covariance matrices. We have used the stepwise version of Stepdisc which alternates forward and backward selection. It is possible to devise an F-ratio which is equivalent to the use of Wilks' statistic. Significance levels were set at the same values as above, Stepdisc selected 14 variables (see Fig. 8). Fig. 17 gives the variables selected and the values for F and the associated confidence value.

Appendix 2. Saliency

The usual cost function is defined as:

$$C(W, D_m) = \tfrac{1}{2}\sum_{k=1}^{m} \|F_N(x^k) - y^k\|^2 \tag{A.10}$$

Variable	F value	P > F
7	99.168	0.0001
11	91.065	0.0001
17	24.630	0.0001
12	22.907	0.0001
5	8.831	0.0002
19	7.118	0.0010
18	5.796	0.0034
13	4.950	0.0077
4	3.910	0.0211
10	3.195	0.0424
29	3.244	0.0404
38	2.962	0.0533
39	2.364	0.0959
40	2.344	0.0978

Fig. 17. Stepdisc results on the waveforms. $P > F$ is the probability that the F stat is superior to the F value.

To order 2, the cost function C of Eq. (A.10) can be approximated, around a minimum W_0 by:

$$C(W) = C(W_0) + (W - W_0)^t \cdot \nabla C(W_0)$$
$$+ \tfrac{1}{2}(W - W_0)^t \cdot H(W_0) \cdot (W - W_0) + o\left(\| W - W_0 \|^2 \right)$$

or, since W_0 is a minimum

$$C(W) = C(W_0) + \tfrac{1}{2}\sum_i H_{ii} \Delta W_i^2 + \tfrac{1}{2}\sum_{ij} H_{ij} \Delta W_i \Delta W_j + o\left(\| W - W_0 \|^2 \right)$$

where $\Delta W = W - W_0$.

If we assume that the Hessian can be approximated by a diagonal matrix, and that the cost is locally quadratic, then we obtain:

$$C(W) = C(W_0) + \tfrac{1}{2}\sum_i H_{ii} \Delta W_i^2$$

and the *saliency* of weight i is thus:

$$s_i = \tfrac{1}{2} H_{ii} W_i^2 = \tfrac{1}{2} \frac{\partial^2 C}{\partial W_i^2} W_i^2 \tag{A.11}$$

Appendix 3. Error bars

Prediction

Our method to compute error bars is based on a technique introduced in [12]: if one assumes a probability distribution on Neural Network weights, then this induces a probability distribution on the NN outputs. The variance of this distribution provides error bars on the outputs. It is then easy to use those to provide error bars for other measures, such as e.g. MSE, or arv.

Suppose that weights follow a Gaussian distribution, locally around a minimum W_0 (this is a common assumption, see e.g. [23,24]):

$$P(W) = \alpha e^{-\beta \sum_i \frac{1}{2} H_{ii} W_i^2} \tag{A.12}$$

where notations are as in Appendix 2, α and β being hyper-parameters which have to be determined.

Using results in [23,24], the regularizing hyper-parameter β can be estimated by:

$$\beta^* = \frac{mQ}{2C\left(W_0, D_m^l\right)} \tag{A.13}$$

Q is the output dimension; in this paper, we have $Q = 1$.

Then, assuming that the output is linear (as is our case in this paper), [12] show that, for every input x^k, the outputs follow a Gaussian with mean $F_N(x^k)$ and variance α_{jk}^2 given by:

$$\alpha_{jk}^2 = \sum_i \frac{\gamma_{ji}^2(k)}{\beta H_{ii}} \quad j = 1, \dots, Q \tag{A.14}$$

where $\gamma_{ji}(k)$ is the gradient of the output $F_N(x^k)$ with respect to weight W_i, and the Hessian is evaluated for input vector x^k. The error bar on output j is thus α_{jk}.

It then follows that the MSE on data set can be bounded in the interval:

$$I(D) = [MSE - A_-, MSE + A_+] \tag{A.15}$$

where:

$$A_- = \frac{1}{m} \sum_{(x^k, y^k) \in D} \left| F_N(x^k) - y^k \right| \alpha_k$$

$$A_+ = A_- + \frac{1}{2m} \sum_{(x^k, y^k) \in D} \alpha_k^2$$

Net	Set	MSE	MSE - A_-	MSE + A_+
12-1	D_m^l	0.04461	0.04439	0.04484
	D_m^v	0.04629	0.04606	0.04653
	D_m^t	0.04209	0.04188	0.04231
10-7-1	D_m^l	0.04359	0.04311	0.04405
	D_m^v	0.05049	0.05004	0.05101
	D_m^t	0.04672	0.04625	0.04720
3-7-1	D_m^l	0.04553	0.04524	0.04582
	D_m^v	0.04851	0.04822	0.04881
	D_m^t	0.04533	0.04505	0.04562
4-7-1	D_m^l	0.04495	0.04463	0.04527
	D_m^v	0.04790	0.04758	0.04823
	D_m^t	0.04488	0.04456	0.04521
5-7-1	D_m^l	0.04465	0.04430	0.04500
	D_m^v	0.04832	0.04796	0.04867
	D_m^t	0.04451	0.04417	0.04486

Fig. 18. Error bars on MSE for the various networks found in the paper.

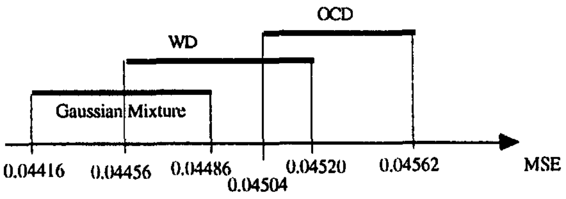

Fig. 19. Confidence intervals for the networks retrained with the variables subsets, selected by OCD – or stepwise – Regularization with weight decay or with a Gaussian mixture.

where α_k is the expression given in (A.14), where index j has been dropped, since we only have one output ($Q = 1$).

These error-bars are not exact: they rely on various assumptions made along the above derivation. However, we have usually found these error bars correct on prediction problems [21].

We give below (Fig. 18) the various intervals found for the networks described in the paper.

These error bars demonstrate (Fig. 19) that:

- variable selection by OCD – or the stepwise technique – is significantly worse than by Regularization with a Gaussian mixture.
- variable selection by OCD – or the stepwise technique – is not significantly worse than by Regularization with Weight Decay.
- variable selection by Regularization with a Gaussian mixture is not significantly better than by Regularization with Weight Decay. The two priors on weights lead to similar results.

Classification

Let p be the classification error and f the observed frequency, p can be considered as a proportion for a binomial law. Under a normal assumption, the 1-α confidence interval $P(p_1 < p < p_2) = 1 - \alpha$ is given by:

$$p_1 = \frac{2f + \dfrac{u_{\alpha/2}^2}{N} + u_{\alpha/2}\sqrt{\dfrac{4f(1-f)}{N} + \dfrac{u_{al2}^2}{N^2}}}{2\left(1 + \dfrac{u_{\alpha/2}^2}{N}\right)}$$

$$\text{and } p_2 = \frac{2f + \dfrac{u_{\alpha/2}^2}{N} - u_{\alpha/2}\sqrt{\dfrac{4f(1-f)}{N} + \dfrac{u_{\alpha/2}^2}{N^2}}}{2\left(1 + \dfrac{u^2\alpha/2}{N}\right)}$$

	$1-\alpha=0.80$; $u_{\alpha/2}=1.28$				$1-\alpha=0.90$; $u_{\alpha/2}=1.64$				$1-\alpha=0.95$; $u_{\alpha/2}=1.96$			
Performance level	80.0	81.0	82.0	83.0	80.0	81.0	82.0	83.0	80.0	81.0	82.0	83.0
Training set	0.83	0.84	0.85	0.86	0.84	0.84	0.85	0.86	0.84	0.85	0.86	0.87
# 300	0.77	0.78	0.80	0.80	0.76	0.77	0.78	0.79	0.75	0.76	0.77	0.78
Validation set	0.82	0.83	0.840	0.84	0.82	0.83	0.84	0.85	0.82	0.83	0.84	0.850
# 1000	0.78	0.79	.80	0.81	0.78	0.79	0.80	0.81	0.77	0.78	0.79	.81
Test set	0.81	0.82	0.83	0.84	0.81	0.82	0.83	0.84	0.81	0.82	0.83	0.84
# 5000	0.79	0.80	0.81	0.82	0.79	0.80	0.81	0.82	0.79	0.80	0.81	0.82

Fig. 20. Confidence intervals for the waveforms classification problem. # nb gives the size of the corresponding set.

where N is the population size and $u_{\alpha/2}$ is the fractil of the normal law at risk $\alpha/2$. Values of p_1, p_2 for the different set sizes are given in Fig. 20.

References

[1] H. Akaike, A new look at the statistical model identification, *IEEE Trans. Auto. Control* 19 (1974) 716–723.

[2] R. Battiti, Using mutual information for selecting features in supervised neural net learning, *IEEE Trans. NN* 5 (4) (1994).

[3] C. Bishop, Exact calculation of the Hessian matrix for the multilayer perceptron, *Neural Comp.* 4 (1992) 494–501.

[4] M. Bollivier de, P. Gallinari and S. Thiria, Cooperation of neural nets and task decomposition, *IJCNN'91*, Seattle, vol. II (1991) 573–576.

[5] G.E.P. Box and G.C. Tiao, *Bayesian Inference in Statistical Analysis* (Addison Wesley, 1973).

[6] L. Breiman, J. Freidman, R. Olshen and C. Stone, *Classification and Regression Trees* (Wadsworth Int. Group 1984).

[7] W.L. Buntine and A.S. Weigend, Bayesian back-propagation, *Complex systems* 5 (1991).

[8] Y. Chauvin, A back-propagation algorithm with optimal use of hidden units, in: *Neural Information Processing Systems, NIPS'88*, D. Touretzky ed., vol. 1 (Morgan Kaufmann, 1989) 519–526.

[9] Y. Chauvin, Dynamic behavior of constrained back-propagation networks, in: *Neural Information Processing Systems*, Denver 1989, D. Touretzky ed. (Morgan Kaufmann, 1990) vol. 2, 643–649.

[10] T. Cibas, F. Fogelman Soulie, P. Gallinari and S. Raudys, *Variable Selection with Neural Networks, ICANN'94, Proceedings*, M. Marinaro and P.G. Morasso eds (Springer-Verlag, 1994) vol. 2, 1464–1469.

[11] T. Cibas, F. Fogelman Soulie, P. Gallinari and S. Raudys, *Variable Selection with Optimal Cell Damage. ICANN'94, Proceedings*, M. Marinaro and P.G. Morasso eds (Springer-Verlag, 1994) vol. 1, 727–730.

[12] J.S. Denker and Y. Le Cun, Transforming neural net output levels to probability distributions, in *Advances in Neural Information Processing Systems*, R. Lippmann, J. Moody and D. Touretzky eds., vol. 3 (Morgan Kaufmann, 1991) 853–859.

[13] R.O. Duda and P.E. Hart, *Pattern Recognition and Scene Analysis* (Wiley, 1973).

[14] W. Finnoff, F. Hergert and H.G. Zimmermann, Improving model selection by nonconvergent methods, *Neural Networks* 6 (6) (1993) 771–783.

[15] K. Fukunaga, *Statistical Pattern Recognition*, 2nd ed. (Academic Press, 1990).

[16] J.D.F. Habema and J. Hermans, Selection of variables in discriminant analysis by F-statistic and error rate, *Technometrics* 19 (4) (1977).

[17] S.J. Hanson and L.Y. Pratt, Comparing biases for minimal network construction with back-propagation in: *Neural Information Processing Systems, NIPS'89*, D.S. Touretzky ed., (Morgan Kaufmann, 1989) vol. 1, 177–185.

[18] B. Hassibi and D.G. Stork, Second order derivatives for network pruning: Optimal brain surgeon, In *Neural Information Processing Systems, NIPS'92*, S.J. Hanson, J.D. Cowan and C.L. Giles eds. (Morgan Kaufmann, 1993) vol. 5, 164–171.

[19] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach* (Prentice-Hall, 1982).

[20] E.D. Karnin, A simple procedure for pruning back-propagation trained nural networks, *IEEE Trans. NN* 1 (2) (1990) 239–242.

[21] A. Kouam, Approches connexionnistes pour la prévision des séries temporelles, Thèse, Université de Paris-Sud, 1993.

[22] Y. Le Cun, J.S. Denker and S.A. Solla, Optimal brain damage, in: *Neural Information Processing Systems, NIPS'89*, D. Touretzky ed. (Morgan Kaufmann, 1990) vol. 2, 598–605.

[23] D.J.C. McKay, A practical bayesian framework for backpropagation networks, *Neural Computation* 4 (1992) 448–472.

[24] D.J.C. McKay, Bayesian interpolation, *Neural Computation* 4 (1992) 415–447.

[25] D.J.C. McKay, Bayesian non-linear modeling for the energy prediction competition, Tech. Rep., University of Cambridge, 1993.

[26] A.J. Miller, *Subset Selection in Regression* (Chapman and Hall, 1990).

[27] T.J. Mitchell and J.J. Beauchamp, Bayesian variable selection in linear regression, *JASA* 83, (1988) 1023–1036.

[28] M. Moller, Efficient training of feed-forward neural networks, Thesis, Univ. Aarhus, 1993.

[29] M.C. Mozer and P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevance assesment, *NIPS* 1 (1989) 107–115.

[30] S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight-sharing, *Neural Computation* 4 (1992) 473–493.

[31] F. Girosi, M. Jones and T. Poggio, Regularization theory and neural networks architectures, *Neural Computation* 7 (2) (1995) 219–269.

[32] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation in *Parallel Distributed Processing*. D.E. Rumelhart and J.L. McClelland eds., (MIT Press, 1986) vol. 1, 318–362.

[33] SAS/Stat User's Guide, version 6, 4th ed.

[34] G. Schwarz, Estimating the dimension of a model. *Annals of Statistics*, 6–2, (1978) 461–464.

[35] H.H. Thodberg, Ace of Bayes; application of neural networks with pruning, Tech. Rep. n°1132E, Danish Meat Res. Inst., 1993.

[36] M.L. Thompson, Selection of variables in multiple regression: Part 1 & 2, *Int. Stat. Rev*, 46 (1978) 1–19 & 129–146.

[37] A.S. Weigend, D.E. Rumelhart and B.A. Huberman, Generalization by weight elimination with application to forecasting, In: *Neural Information Processing Systems, NIPS'90*. R.P. Lippmann, J.E. Moody and D.S. Touretzky eds., (Morgan Kaufmann, 1991) vol. 3, 875–882.

Tautvydas Cybas got an M.S. degree in applied mathematics from university of Vilnius (Lt) in 1990. He then joined the Institute of Mathematics and Informatics at Vinius and is currently working through his Ph.D. at the University of Orsay (Fr). His main interests are in statistical aspects of neural networks.

Francoise Fogelman Soulie got her Ph.D. in Computer Science from the University of Grenoble. A former professor at the University of Paris-Sud-Orsay, she co-founded Mimetics-sa, a start-up in neural networks and later-on joined Sligos-sa, where she is now head of the Pattern Recognition Laboratory. Her research interests are in developing neural networks algorithms for image processing, optical character recognition and time series prediction. She is a member of IEEE, of the Board of Governors of INNS and of ENNS. She is a member of the Scientific Committee of France Telecom, an expert for the European Commission and Chevalier des Palmes Academiques.

Patrick Gallinari received the PhD degree from the University of Compiegne (Fr) in 1986. He is Professor at Laboratoire Formes et Intelligence Artificielle, University Paris 6 (Fr). His main research interests are in neural networks, pattern recognition, symbolic-numerical systems. He is a member of the board of ENNS.

Sarunas J. Raudys was born in Kaunas, Lithuania, on February 24, 1941. He received the M.S. degree in electrical and computer engineering from Kaunas University of Technology in 1963, and the Candidate of Sciences and Doctor of Sciences degrees from the Institute of Mathematics and Cybernetics, Academy of Sciences, Lithuania, in 1969 and 1978, respectively. He is currently a Head of the Data Analysis Department in the Institute of Mathematics and Informatics, Vilnius, Lithuania. His current research interests include multivariate statistical analysis, pattern recognition, artificial neural nets, machine learning and data analysis methods. Prof. Raudys is a member of the New York Academy of Sciences, an Associate Editor of *Pattern Recognition* (J. of Int. Pattern Recognition Society, Washington, DC) and *Pattern Recognition and Image Processing* (J. of Russian Academy of Sciences, Moscow). He has been a member of the Program Committee and an invited speaker in a number of international conferences.