Variable selection with neural networks

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

a LRI, bát. 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

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
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 \(D_m = \{(x_1^1, y_1^1), \ldots, (x_m^m, y_m^m)\}\), 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:

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

(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:

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

(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:

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

(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:

$$M(W) = \alpha C(W, D_m) + \beta E_W(W)$$

(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]:

$$E_W(W) = \sum_j W_j^2$$

(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:

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

(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):

$$\ln P(W) = -|W| \ln \sigma_{W^2} - \frac{1}{2\sigma_W^2} \sum_j W_j^2$$

(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 $\alpha$ and $\beta$ 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 hyperparameters 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 \(m_t; D^l_m, \) for
learning, \(D^v_m, \) for validation (i.e. set the parameters) and \(D^t_m, \) for test. Our training
procedure is an on-line GBP implemented as follows: the gradient step \(\varepsilon\) is started
from an initial value \(\varepsilon_0\) and decreased as soon as the relative average cost variation
becomes too small. It has been chosen here for its simplicity.

\[
\text{GBP} \quad \bullet \text{run N steps of on-line GBP with } \varepsilon = \varepsilon_0. \\
\bullet \text{while } \varepsilon \geq \varepsilon_1 \text{ do} \\
\quad \text{run on-line GBP on } D^l_m \text{ with } \varepsilon \\
\quad \text{if } r(W, D^v_m, t, T) \leq \theta_1 \\
\qquad \text{then } \varepsilon(t+1) = \alpha \varepsilon(t) \text{ else } \varepsilon(t+1) = \varepsilon(t) \\
\text{endwhile}
\]

where

\[
\begin{align*}
r(W, D, t, T) &= 1 - \frac{[C(W, D)_T]_t}{[C(W, D)]_{t-1}^T} \\
[C(W, D)]_T^T &= \frac{1}{T} \sum_{\tau=0}^{T-1} C(W(t-\tau), D)
\end{align*}
\]
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00

Fig. 1. Time series $z_t$ from validation set $D^m$. 

$r(W, D, t, T)$ 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: $e_0$, $e_1$ and $\theta_1$ being given, $N$, $T_1$ and $\alpha$ 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:

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

$$y_t = x_t + 0.3x^2_{t-6} - 0.2x^2_{t-4}$$

(3.2)

where $u_t$ is a white noise.

Three independent samples of size $m = 1000$ each are drawn from series $y_t$: $D^l_m$, $D^u_m$ and $D^r_m$. 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 $D^l_m$, $D^u_m$ and $D^r_m$.

Fig. 1 shows set $D^u_m$.

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.
0.04072 (after normalization of y). 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 \(x_{t-10}\) to \(x_{t-1}\) as inputs, 7 non linear hidden units and 1 linear output). Now, selection should pick up input variables \((x_{t+5}, x_{t+1}, x_{t-1})\).

Results are shown in Fig. 2 for net 10-7-1 and net 12-1. We use the MSE on \(D_m^r\) to choose the best parameter values for net 10-7-1 (for the sake of simplicity, only results for choosing \(T_1\) and \(\alpha\) 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 \(\epsilon_0 = 0.1, \epsilon_1 = 0.0001, \theta_1 = 0.0005.\) Values \(\epsilon_0 = 0.1, \epsilon_1 = 0.000005, \theta_1 = 0.000025\) (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 \(D_m^r\) being very oscillating, and \(D_m^l\) much smoother. This explains why performances on test set \(D_m^l\) were often found better than on validation set \(D_m^r\).

### 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:

\[
x_i = \frac{uH^{m}_i + (1-u)H^{n}_i}{5} + \epsilon_i, \quad 0 \leq i \leq 20
\]

\[
x_i = \epsilon_i, \quad 21 \leq i \leq 40
\]
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.

Fig. 4. Final MSE for waveforms, net 40-10-3 (top) and 40-3 (bottom).
where \( x_i \) denotes the \( i \)th component of a pattern \( x \), \( u \) is a uniform random variable in \([0, 1]\), \( \varepsilon_i \) 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 \( x \). \( D_m^t \) has 300 elements, \( D_m^v 1000 \) and \( D_m^w 5000 \).

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: \( T_1 = 10, \alpha = 0.85, \varepsilon_0 = 0.2, \varepsilon_1 = 0.0001, \theta_0 = 0.0001 \) (net 40-10-3) and \( T_1 = 30, \alpha = 0.95, \varepsilon_0 = 0.2, \varepsilon_1 = 0.0001, \theta_1 = 0.0001 \) (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:

\[
\zeta_i = \sum_{j \in \text{Fan\_Out}(i)} s_j
\]

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.

\[
\text{OCD} \quad \text{if} \quad r(W, D_m^v, t - 1, T_1) \leq \theta_2 \quad \text{and} \quad r(W, D_m^v, t, T_1) \leq \theta_2 \quad \text{then} \quad \begin{align*}
\text{compute saliencies } \zeta_i \text{ at time } t \text{ and order them: } \zeta_i_1 \geq \zeta_i_2 \geq \ldots \geq \zeta_i_p \\
\text{choose } p \text{ such that: } \frac{\sum_{i=1}^{p} \zeta_{ii}}{\sum_{i=1}^{p} \zeta_{ii}} \geq q \text{ and eliminate variables } i_{p+1}, \ldots, i_{p} \\
\text{continue OCD} \\
\end{align*}
\]

\[
\text{else continue GBP}
\]

Pruning starts early if \( \theta_2 \) is large, late otherwise. Both parameters \( \theta_2 \) 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.
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).
Fig. 8. Selected variables for the different selection methods on Waveforms. M-X: method M on network X (X = 40-10-3 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.

![Table Image](image)

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.
4.3. Waveforms

For the waveforms, we ran our OCD algorithm with values of $T_1$ and $\alpha$ 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). $y_i$ 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:

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

$$\ln P(V_b, V, W) \propto \frac{1}{2\sigma^2} \sum_{i,j} V_{ji}^2$$  \hspace{1cm} (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

- 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 $r(W, D_m, t, T_1) \leq -\theta_1$
    - 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 $\alpha$ as set by cross-validation after GBP (see Section 3.2.), for the given values of $\varepsilon_0$, $\varepsilon_1$ and $\theta_1$. The first $N$ steps are exactly as before; from $N$ on, regularization starts.

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

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

- order variances: $\var_{i_1} \geq \var_{i_2} \geq \ldots \geq \var_{i_p}$
- choose smallest $p$ such that: $\sum_{k=1}^{p} \var_{i_k} \geq q \sum_{k=1}^{p} \var_{i_k}$
Fig. 11. Weight Decay on Time series: determination of $\lambda^*$ 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.

Fig. 12. Weight Decay on Waveforms: determination of $\lambda^*$ 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, is Gaussian N(0, $\sigma_w^2$), resp. N(0, $\sigma_v^2$). $V_j$'s are supposed to be independently distributed with a Gaussian mixture of two components: one – in proportion $\pi$ - N(0, $\sigma_1^2$) has a large variance $\sigma_1^2$ and the other – in proportion $(1 - \pi)$ - N(0, $\sigma_2^2$) a small one $\sigma_2^2$. Under these assumptions, we have:

$$\ln P(V_b, V, W) \propto -\frac{1}{2\sigma_w^2} \sum \sum W_{sj}^2 - \frac{1}{2\sigma_v^2} \sum V_{bj}^2 + \sum_1^P \ln \left[ \frac{\pi}{\sigma_1^2} + \frac{1 - \pi}{\sigma_2^2} \right]$$

$$M(V_b, V, W)$$ is minimized, through an on-line GBP (see Section 3). We have to determine the following parameters: $\sigma_w^2$, $\sigma_1^2$, $\sigma_2^2$, $\pi$, $q$ and $\alpha^2$. 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.

![Fig. 13. Time series: determination of $\pi^*$ with Regularization (with Gaussian mixture) and of $q^*$ with Variable selection.](image-url)
5.3.2. Time series

In this paragraph, all hyperparameters are held constant \((\sigma^2_W = \sigma^2_V = 1, \sigma^2_h = 0.005)\), while weights \(V\) and \(W\) are updated; \(\tau, \sigma^2\) (in 5.1) are set by cross-validation during algorithm Regularization: only results with the best \(\sigma^2\) 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 \(\tau\) and \(\sigma^2\) are thus fixed; \(\sigma^2_t, \sigma^2_r\) (and \(\sigma^2_h\) 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 of 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

<table>
<thead>
<tr>
<th>Set</th>
<th>Method</th>
<th>Cross-validation</th>
<th>Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D^0_m)</td>
<td>0.04585</td>
<td>0.04485</td>
<td></td>
</tr>
<tr>
<td>(D^0_m)</td>
<td>0.04820</td>
<td>0.04750</td>
<td></td>
</tr>
<tr>
<td>(D^0_m)</td>
<td>0.04556</td>
<td>0.04378</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 14. Time series, final MSE reached after Regularization, when parameters are set by cross-validation (left) or estimated (right).
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 $\sigma_W^2 = \sigma_t^2 = 1$ fixed a priori. We then train $N$ steps with cost function $C$. We then use as an estimate for $\sigma^2$ the MSE on $D^w_n$, a new estimate being computed after each epoch. $\sigma^2_t$ 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. $\sigma^2, \sigma_W^2, \sigma_t^2, \sigma^2_t$).
- 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:

\[
F = \frac{m - k}{k - q} \frac{\| \hat{Y}_k - \hat{Y}_q \|^2}{\| Y - \hat{Y}_k \|^2}.
\]

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

\[
\hat{Y}_i = \left( F_{\alpha}(X^1), \ldots, F_{\alpha}(X^m) \right)^t
\]

Let \( \theta_\alpha \) be given by:

\[
P(F > \theta_\alpha) = \alpha
\]

where \( F \) is a Fisher-Snedecor variable with \((k - q, m - k)\) degrees of freedom and \( \alpha \) is some confidence level (e.g. \( \alpha = 0.95 \)). Then, if the measured ratio \( F \) (A.1) is larger than \( \theta_\alpha \), 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.

\[
\begin{align*}
F > \theta_\alpha & \quad \text{accept } H_0: \text{use all } k \text{ variables} \\
F < \theta_\alpha & \quad \text{reject } H_0: \text{use } q \text{ variables, reject the other } k - q \text{ variables}
\end{align*}
\]

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:

\[
J_{k_1} \leq J_{k_2} \leq \cdots \leq J_{k_N}
\]

where \( k_k \) 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 \( k_k \) is the feature vector at instant \( k \), a feature \( x_{k+1} \) is selected in \( X - k_k \) so that:

\[
J(k_k \cup \{x_{k+1}\}) = \max_{x_{k+1} \in X - k_k} J(k_k \cup \{x_{k+1}\})
\]

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:

\[
J(k_{N-k} \setminus \{x_{k+1}\}) = \max_{x_{k+1} \in k_{N-k}} J(k_{N-k} \setminus \{x_{k+1}\})
\]

where \( k_{N-k} \) is the feature vector at iteration \( k \).
The $F$-factor can be used as a criterion $J$; Eqs. (A.6) and (A.7) then become:

$$F_i = (m - k - 1) \frac{\| \hat{Y}_{k+1}^i - \hat{Y}_k^i \|^2}{\| Y - \hat{Y}_{k+1}^i \|^2}, \quad x_{k+1} = \text{Arg max}_{x_i \in X \setminus \mathcal{K}_k} F_i$$  \hspace{1cm} (A.8)

where $\mathcal{K}_k \subset \mathcal{K}_{k+1} = \mathcal{K}_k \cup \{ x_i \}$ and $\hat{Y}_k^i$, resp. $\hat{Y}_{k+1}^i$, is the estimator produced using variables in $\mathcal{K}_k$, resp. $\mathcal{K}_{k+1}$.

$$F_i = (m - N + k) \frac{\| \hat{Y}_{N-k}^i - \hat{Y}_{N-k-1}^i \|^2}{\| Y - \hat{Y}_{N-k}^i \|^2}, \quad x_{k+1} = \text{Arg max}_{x_i \in \mathcal{K}_{N-k}} F_i$$  \hspace{1cm} (A.9)

where $\mathcal{K}_{N-k} \supset \mathcal{K}_{N-k-1} = \mathcal{K}_{N-k} \setminus \{ x_i \}$ and $\hat{Y}_{N-k}^i$, resp. $\hat{Y}_{N-k-1}^i$, is the estimator produced using variables in $\mathcal{K}_{N-k}$, resp. $\mathcal{K}_{N-k-1}$.

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 $\lambda_w$: 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:

$$C(W, D_m) = \frac{1}{2} \sum_{k=1}^{m} \| F_N(x^k) - y^k \|^2$$  \hspace{1cm} (A.10)
To order 2, the cost function $C$ of Eq. (A.10) can be approximated, around a minimum $W_0$ by:

$$C(W) = C(W_0) + (W - W_0)' \cdot \nabla C(W_0) + \frac{1}{2} (W - W_0)' \cdot H(W_0) \cdot (W - W_0) + o(\|W - W_0\|^2)$$

or, since $W_0$ is a minimum

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

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:

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

and the saliency of weight $i$ is thus:

$$s_i = \frac{1}{2} H_{ii} W_i^2 = \frac{\partial^2 C}{\partial W_i^2} W_i^2$$  \hfill (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]):

$$P(W) = \alpha e^{-\beta \sum_i \frac{1}{2} H_{ii} W_i^2}$$  \hfill (A.12)
where notations are as in Appendix 2, \( \alpha \) and \( \beta \) being hyper-parameters which have to be determined.

Using results in [23,24], the regularizing hyper-parameter \( \beta \) can be estimated by:

\[
\beta^* = \frac{mQ}{2C(W_0, D_m^l)}
\]  

(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 \( \alpha^2_{jk} \) given by:

\[
\alpha^2_{jk} = \frac{\gamma^2_{ji}(k)}{\beta H_{ii}^j} \quad j = 1, \ldots, Q
\]  

(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 \( \alpha_{jk} \).

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

\[
I(D) = [MSE - A_-, MSE + A_+]
\]  

(A.15)

where:

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

\[
A_+ = A_- + \frac{1}{2m} \sum_{(x^k,y^k) \in D} \alpha^2_k
\]

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

Fig. 18. Error bars on MSE for the various networks found in the paper.
where $\alpha_k$ 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 $1-\alpha$ confidence interval $P(p_1 < p < p_2) = 1 - \alpha$ is given by:

\[
p_1 = \frac{2f + \frac{u^2_{\alpha/2}}{N} + u_{\alpha/2} \sqrt{\frac{4f(1-f)}{N}} + \frac{u^2_{\alpha/2}}{N^2}}{2 \left(1 + \frac{u^2_{\alpha/2}}{N}\right)}
\]

and

\[
p_2 = \frac{2f + \frac{u^2_{\alpha/2}}{N} - u_{\alpha/2} \sqrt{\frac{4f(1-f)}{N}} + \frac{u^2_{\alpha/2}}{N^2}}{2 \left(1 + \frac{u^2_{\alpha/2}}{N}\right)}
\]
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


Tautvydas Cyhas 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 Vilnius and is currently working through his Ph.D. at the University of Orsay (Fr). His main interests are in statistical aspects of neural networks.
Françoise 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 Académiques.

Patrick Gallinari received the PhD degree from the University of Compiègne (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.