GEOPHYSICS, VOL. 65, NO. 4 (JULY-AUGUST 2000); P. 1032–1047, 7 FIGS., 1 TABLE.

Neural networks in geophysical applications

Mirko van der Baan* and Christian Jutten‡

ABSTRACT

Neural networks are increasingly popular in geophysics. Because they are universal approximators, these tools can approximate any continuous function with an arbitrary precision. Hence, they may yield important contributions to finding solutions to a variety of geophysical applications.

However, knowledge of many methods and techniques recently developed to increase the performance and to facilitate the use of neural networks does not seem to be widespread in the geophysical community. Therefore, the power of these tools has not yet been explored to their full extent. In this paper, techniques are described for faster training, better overall performance, i.e., generalization, and the automatic estimation of network size and architecture.

INTRODUCTION

Neural networks have gained in popularity in geophysics this last decade. They have been applied successfully to a variety of problems. In the geophysical domain, neural networks have been used for waveform recognition and first-break picking (Murat and Rudman, 1992; McCormack et al., 1993); for electromagnetic (Poulton et al., 1992), magnetotelluric (Zhang and Paulson, 1997), and seismic inversion purposes (Röth and Tarantola, 1994; Langer et al., 1996; Calderón–Macías et al., 1998); for shear-wave splitting (Dai and MacBeth, 1994), well-log analysis (Huang et al., 1996), trace editing (McCormack et al., 1993), seismic deconvolution (Wang and Mendal, 1992; Calderón–Macías et al., 1997), and event classification (Dowla et al., 1990; Romeo, 1994); and for many other problems.

Nevertheless, most of these applications do not use more recently developed techniques which facilitate their use. Hence, expressions such as "designing and training a network is still more an art than a science" are not rare. The objective of this paper is to provide a short introduction to these new techniques. For complete information covering the whole domain of neural networks types, refer to excellent reviews by Lippmann (1987), Hush and Horne (1993), Hérault and Jutten (1994), and Chentouf (1997).

The statement that "designing and training a network is still more an art than a science" is mainly attributable to several well-known difficulties related to neural networks. Among these, the problem of determining the optimal network configuration (i.e., its structure), the optimal weight distribution of a specific network, and the guarantee of a good overall performance (i.e., good generalization) are most eminent. In this paper, techniques are described to tackle most of these well-known difficulties.

Many types of neural networks exist. Some of these have already been applied to geophysical problems. However, we limit this tutorial to static, feedforward networks. Static implies that the weights, once determined, remain fixed and do not evolve with time; feedforward indicates that the output is not feedback, i.e., refed, to the network. Thus, this type of network does not iterate to a final solution but directly translates the input signals to an output independent of previous input.

Moreover, only supervised neural networks are considered—in particular, those suited for classification problems. Nevertheless, the same types of neural networks can also be used for function approximation and inversion problems (Poulton et al., 1992; Röth and Tarantola, 1994). Supervised classification mainly consists of three different stages (Richards, 1993): selection, learning or training, and classification. In the first stage, the number and nature of the different classes are defined and representative examples for each class are selected. In the learning phase, the characteristics of each individual class must be extracted from the training examples. Finally, all data can be classified using these characteristics.

Nevertheless, many other interesting networks exist—unfortunately, beyond the scope of this paper. These include the self-organizing map of Kohonen (1989), the adaptive resonance theory of Carpenter and Grossberg (1987), and the Hopfield network (Hopfield, 1984) and other recurrent

Manuscript received by the Editor January 20, 1999; revised manuscript received February 3, 2000.
*Formerly Université Joseph Fourier, Laboratoire de Géophysique Interne et Tectonophysique, BP 53, 38041 Grenoble Cedex, France; currently University of Leeds, School of Earth Sciences, Leeds LS2 9JT, UK. E-mail: mvdbaan@earth.leeds.ac.uk.
‡Laboratoire des Images et des Signaux, Institut National Polytechnique, 46 av. Félix Viallet, 38031 Grenoble Cedex, France. E-mail: chris@lis-viallet.inpg.fr.

networks. See Lippmann (1987) and Hush and Horne (1993) for a partial taxonomy.

This paper starts with a short introduction to two types of static, feedforward neural networks and explains their general way of working. It then proceeds with a description of new techniques to increase performance and facilitate their use. Next, a general strategy is described to tackle geophysical problems. Finally, some of these techniques are illustrated on a real data example—namely, the detection and extraction of reflections, ground roll, and other types of noise in a very noisy common-shot gather of a deep seismic reflection experiment.

NEURAL NETWORKS: STRUCTURE AND BEHAVIOR

The mathematical perceptron was conceived some 55 years ago by McCulloch and Pitts (1943) to mimic the behavior of a biological neuron (Figure 1a). The biological neuron is mainly composed of three parts: the dendrites, the soma, and the axon. A neuron receives an input signal from other neurons connected to its dendrites by synapses. These input signals are attenuated with an increasing distance from the synapses to the soma. The soma integrates its received input (over time and space) and thereafter activates an output depending on the total input. The output signal is transmitted by the axon and distributed to other neurons by the synapses located at the tree structure at the end of the axon (Hérault and Jutten, 1994).

The mathematical neuron proceeds in a similar but simpler way (Figure 1b) as integration takes place only over space. The weighted sum of its inputs is fed to a nonlinear transfer function (i.e., the activation function) to rescale the sum (Figure 1c). A constant bias θ is applied to shift the position of the activation function independent of the signal input. Several examples of such activation functions are displayed in Figure 1d.

Historically, the Heaviside or hard-limiting function was used. However, this particular activation function gives only a binary output (i.e., 1 or 0, meaning yes or no). Moreover, the optimum weights were very difficult to estimate since this particular function is not continuously differentiable. Thus, e.g., first-order perturbation theory cannot be used. Today, the sigmoid is mostly used. This is a continuously differentiable, monotonically increasing function that can best be described as a smooth step function (see Figure 1d). It is expressed by $f_s(\alpha) = (1 + e^{-\alpha})^{-1}$.

To gain some insight in the working of static feedforward networks and their ability to deal with classification problems, two such networks will be considered: one composed of a single neuron and a second with a single layer of hidden neurons. Both networks will use a hard-limiting function for simplicity.

Figure 2a displays a single neuron layer. Such a network can classify data in two classes. For a 2-D input, the two distributions are separated with a line (Figure 2b). In general, the two

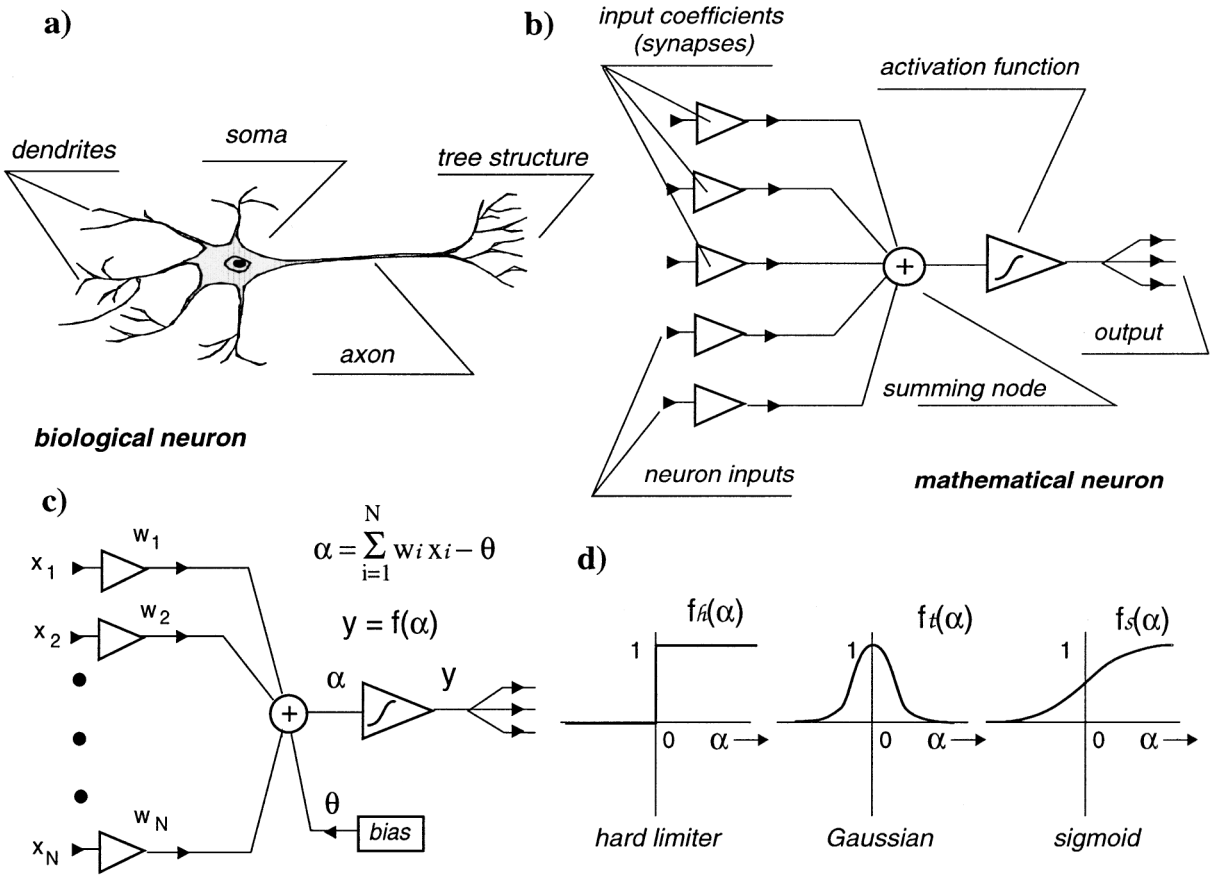

FIG. 1. The biological and the mathematical neuron. The mathematical neuron (b) mimics the behavior of the biological neuron (a). The weighted sum of the inputs is rescaled by an activation function (c), of which several examples are shown in (d). Adapted from Lippmann (1987), Hérault and Jutten (1994), and Romeo (1994).

classes are separated by an $(n-1)$-dimensional hyperplane for an n-dimensional input.

More complex distributions can be handled if a hidden layer of neurons is added. Such layers lie between the input and output layers, connecting them indirectly. However, the general way of working does not change at all, as shown in Figures 3a and 3b. Again, each neuron in the hidden layer divides the input space in two half-spaces. Finally, the last neuron combines these to form a closed shape or subspace. With the addition of a second hidden layer, quite complex shapes can be formed (Romeo, 1994). See also Figure 14 in Lippmann (1987).

Using a sigmoidal instead of a hard-limiting function does not change the general picture. The transitions between classes are smoothed. On the other hand, the use of a Gaussian activation function implicates major changes, since it has a localized response. Hence, the sample space is divided in two parts. The part close to the center of the Gaussian with large outputs is enveloped by the subspace at its tails showing small output values. Thus, only a single neuron with a Gaussian activation function and constant variance is needed to describe the gray class in Figure 3 instead of the depicted three neurons with hard-limiting or sigmoidal activation functions. Moreover, the Gaussian will place a perfect circle around the class in the middle (if a common variance is used for all input parameters).

This insight into the general way neural networks solve classification problems enables a user to obtain a first notion of the structure required for a particular application. In the case of very complicated problems with, say, skewed, multimodal distributions, one will probably choose an neural networks structure with two hidden layers. However, Cybenko (1989) shows that neural networks using sigmoids are able to approximate asymptotically any continuous function with an arbitrary close precision using only a single nonlinear, hidden layer and linear output units. Similarly, Park and Sandberg (1991) show that, under mild conditions, neural networks with localized activation functions (such as Gaussians) are also universal approximators. Unfortunately, neither theorem is able to predict the

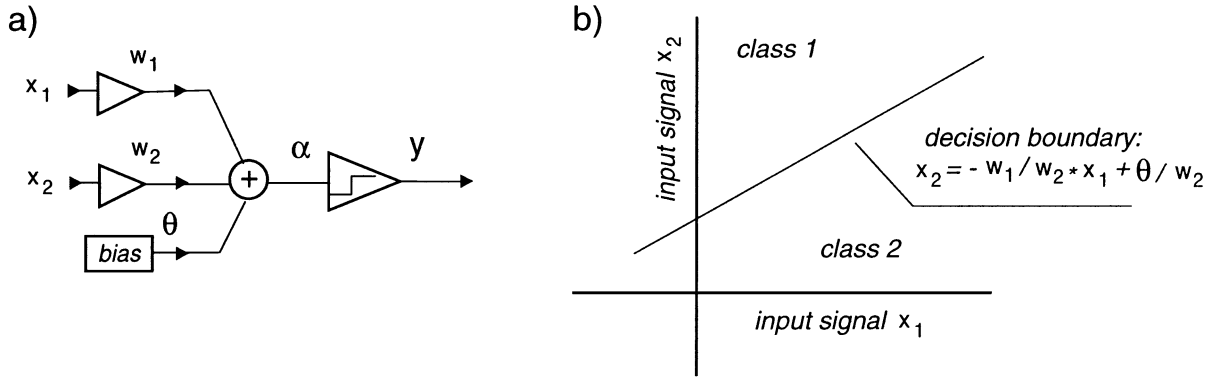

FIG. 2. (a) Single perceptron layer and (b) associated decision boundary. Adapted from Romeo (1994).

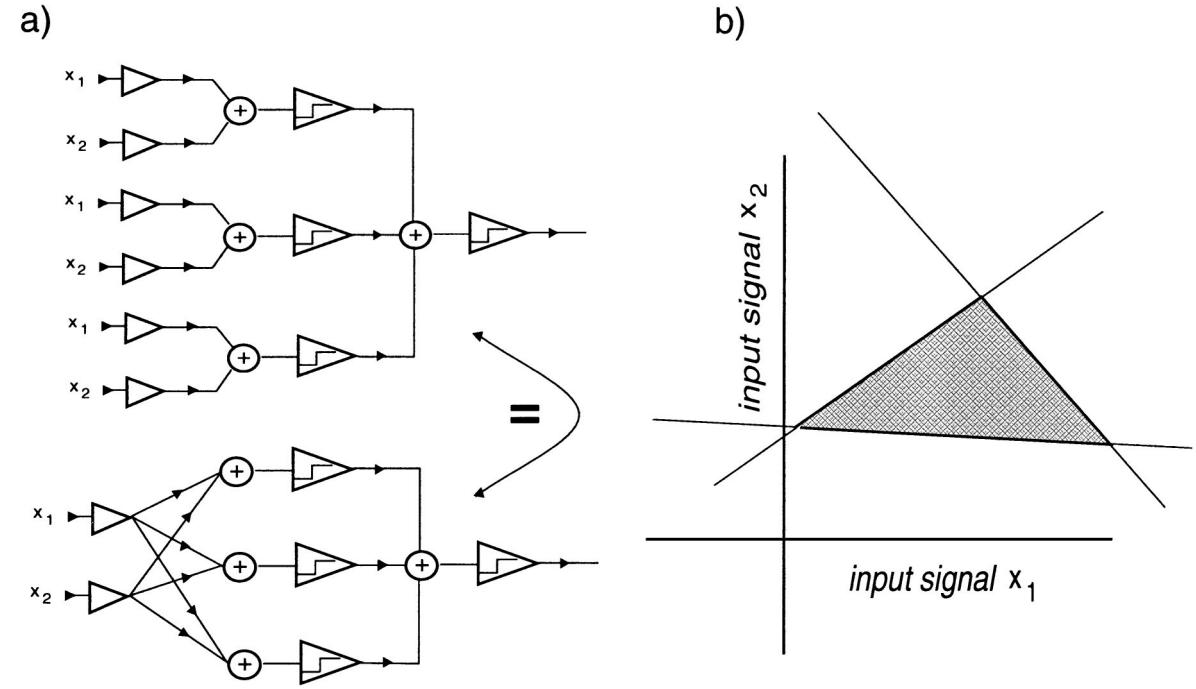

FIG. 3. (a) Single hidden perceptron layer and (b) associated decision boundary. Adapted from Romeo (1994).

exact number of neurons needed since these are asymptotic results. Moreover, applications exist where neural networks with two hidden layers produce similar results as a single hidden layer neural networks with a strongly reduced number of links and, therefore, a less complicated weight optimization problem, i.e., making training much easier (Chentouf, 1997).

Two types of activation functions are used in Figure 1d. The hard-limiter and the sigmoid are monotonically increasing functions, whereas the Gaussian has a localized activation. Both types are commonly used in neural networks applications. In general, neural networks with monotonically increasing activation functions are called multilayer perceptrons (MLP) and neural networks with localized activation functions are called radial basis functions (RBF) (Table 1).

Hence, MLP networks with one output perceptron and a single hidden layer are described by

$$f_{\mathrm{MLP}}(\mathbf{x}) = \sigma\left(\sum_{k=1}^{n_{h1}} w_k \sigma\left(\mathbf{w}^{(k)} \cdot \mathbf{x} - \theta^{(k)}\right) - \theta\right) \qquad (1)$$

with $\sigma(.)$ the sigmoidal activation function, $\mathbf{x}$ the input, w_k the weight of link k to the output node, n_{h1} the number of nodes in the hidden layer, $\mathbf{w}^{(k)}$ the weights of all links to node k in the hidden layer, and θ the biases. Boldface symbols indicate vectors. Equation (1) can be extended easily to contain several output nodes and more hidden layers.

Likewise, RBF networks with a single hidden layer and one output perceptron are described by

$$f_{\mathrm{RBF}}(\mathbf{x}) = \sigma\left(\sum_{k=1}^{n_{h1}} w_k \mathcal{K}\left(s_k \left\|\mathbf{x} - \mathbf{c}^{(k)}\right\|\right) - \theta\right) \qquad (2)$$

with $\mathcal{K}(.)$ the localized activation function, $\|.\|$ a (distance) norm, $\mathbf{c}^{(k)}$ the center of the localized activation function in hidden node k, and s_k its associated width (spread).

It is important to be aware of the total number (n_{tot}) of internal variables determining the behavior of the neural networks structure used, as we show hereafter. Fortunately, this number is easy to calculate from equations. (1) and (2). For MLP networks it is composed of the number of links plus the number of perceptrons to incorporate the number of biases. If n_i denotes the number of input variables, n_{hi} the number of perceptrons in the ith hidden layer, and n_o the number of output perceptrons, then n_{tot} is given by

$$n_{\mathrm{tot}} = (n_i + 1) * n_{h1} + (n_{h1} + 1) * n_o \qquad (3)$$

for an MLP with a single hidden layer and

$$n_{\mathrm{tot}} = (n_i + 1) * n_{h1} + (n_{h1} + 1) * n_{h2} + (n_{h2} + 1) * n_o \qquad (4)$$

for an MLP with two hidden layers. The number of internal variables is exactly equal for isotropic RBF networks since each Gaussian is described by $n_i + 1$ variables for its position and variance, i.e., width. Moreover, in this paper only RBF networks with a single hidden layer are considered. In addition, only RBF neurons in the hidden layer have Gaussian activation functions. The output neurons have sigmoids as activation functions. Hence, n_{tot} is also given by equation (3).

As we will see, the ratio n_{tot}/m determines if an adequate network optimization can be hoped for, where m defines the number of training samples.

Table 1. Abbreviations.

MLP	Multilayer Perceptrons
NCU	Noncontributing Units
OBD	Optimal Brain Damage
OBS	Optimal Brain Surgeon
PCA	Principal Component Analysis
RBF	Radial Basic Functions
SCG	Scaled Conjugate Gradient

NETWORK OPTIMIZATION

Known problems

The two most important steps in applying neural networks to recognition problems are the selection and learning stages, since these directly influence the overall performance and thus the results obtained. Three reasons can cause a bad performance (Romeo, 1994): an inadequate network configuration, the training algorithm being trapped in a local minimum, or an unsuitable learning set.

Let us start with the network configuration. As shown in Figures 2 and 3, the network configuration should allow for an adequate description of the underlying statistical distribution of the spread in the data. Since the number of input and output neurons is fixed in many applications, our main concern is with the number of hidden layers and the number of neurons therein.

No rules exist for determining the exact number of neurons in a hidden layer. However, Huang and Huang (1991) show that the upper bound of number of neurons needed to reproduce exactly the desired outputs of the training samples is on the order of m, the number of training samples. Thus, the number of neurons in the hidden layer should never exceed the number of training samples. Moreover, to keep the training problem overconstrained, the number of training samples should always be larger than the number of internal weights. In practice, $m \approx 10 n_{\mathrm{tot}}$ is considered a good choice. Hence, the number of neurons should be limited; otherwise, the danger exists that the training set is simply memorized by the network (overfitting). Classically, the best configuration is found by trial and error, starting with a small number of nodes.

A second reason why the network may not obtain the desired results is that it may become trapped in a local minimum. The misfit function is very often extremely complex (Hush et al., 1992). Thus, the network can easily be trapped in a local minimum instead of attaining the sought-for global one. In that case even the training set cannot be fit properly.

Remedies are simple. Either several minimization attempts must be done, each time using a different (random or nonrandom) initialization of the weights, or other inversion algorithms must be considered, such as global search.

Finally, problems can occur with the selected training set. The two most frequent problems are overtraining and a bad, i.e., unrepresentative, learning set. In the latter case, either too many bad patterns are selected (i.e., patterns attributed to the wrong class) or the training set does not allow for a good generalization. For instance, the sample space may be incomplete, i.e., samples needed for an adequate training of the network are simply missing.

Overtraining of the learning set may also pose a problem. Overtraining means the selected training set is memorized such that performance is only excellent on this set but not on other data. To circumvent this problem, the selected set of examples

is often split into a training and a validation set. Weights are optimized using the training set. However, crossvalidation with the second set ensures an overall good performance.

In the following subsections, all of these problems are considered in more detail, and several techniques are described to facilitate the use of neural networks and to enhance their performance.

Network training/weight estimation: An optimization problem

If a network configuration has been chosen, an optimal weight distribution must be estimated. This is an inversion or optimization problem. The most common procedure is a so-called localized inversion approach. In such an approach, we first assume that the output $\mathbf{y}$ can be calculated from the input $\mathbf{x}$ using some kind of function $\mathbf{f}$, i.e., $\mathbf{y} = \mathbf{f}(\mathbf{x})$. Output may be contaminated by noise, which is assumed to be uncorrelated to the data and to have zero mean. Next, we assume that the function can be linearized around some initial estimate $\mathbf{x}_0$ of the input vector $\mathbf{x}$ using a first-order Taylor expansion, i.e.,

$$\mathbf{y} = \mathbf{f}(\mathbf{x}_0) + \frac{\partial \mathbf{f}(\mathbf{x}_0)}{\partial \mathbf{x}} \Delta \mathbf{x}. \tag{5}$$

If we write $\mathbf{y}_0 = \mathbf{f}(\mathbf{x}_0)$, $\Delta \mathbf{y} = \mathbf{y} - \mathbf{y}_0$, and $\mathbf{A}^{(x)} = \partial \mathbf{f} / \partial \mathbf{x}$, equation (5) can also be formulated as

$$\Delta \mathbf{y} = \mathbf{A}^{(x)} \Delta \mathbf{x}, \tag{6}$$

where the Jacobian $\mathbf{A}^{(x)} = \nabla_{\mathbf{x}} \mathbf{f}$ contains the first partial derivatives with respect to $\mathbf{x}$. To draw an analogy with a better known inversion problem, in a tomography application $\mathbf{y}$ would contain the observed traveltimes, $\mathbf{x}$ the desired slowness model, and A_{ij} the path lengths of ray i in cell j.

However, there exists a fundamental difference with a tomography problem. In an neural networks application, both the output $\mathbf{y}$ and the input $\mathbf{x}$ are known, since $\mathbf{y}^{(i)}$ represents the desired output for training sample $\mathbf{x}^{(i)}$. Hence, the problem is not the construction of a model $\mathbf{x}$ explaining the observations, but the construction of the approximation function $\mathbf{f}$. Since this function is described by its internal variables, it is another linear system that must be solved, namely,

$$\Delta \mathbf{y} = \mathbf{A}^{(w)} \Delta \mathbf{w}, \tag{7}$$

where the Jacobian $\mathbf{A}^{(w)} = \nabla_{\mathbf{w}} \mathbf{f}$ contains the first partial derivatives with respect to the internal variables $\mathbf{w}$. The vector $\mathbf{w}$ contains the biases and weights for MLP networks and the weights, variances, and centers for RBF networks. For the exact expression of $\mathbf{A}^{(w)}$, we refer to Hush and Horne (1993) and Hérault and Jutten (1994). Nevertheless, all expressions can be calculated analytically. Moreover, both the sigmoid and the Gaussian are continuously differentiable, which is the ultimate reason for their use. Thus, no first-order perturbation theory must be applied to obtain estimates of the desired partial derivatives, implying a significant gain in computation time for large neural networks.

In general, the optimization problem will be ill posed since $\mathbf{A}^{(w)}$ suffers from rank deficiency, i.e., rank $(\mathbf{A}^{(w)}) \leq n_{\text{tot}}$. Thus, system (7) is underdetermined. However, at the same time, any well-formulated inversion problem will be overconstrained because $m \gg n_{\text{tot}}$, yielding that there are more training samples than internal variables.

Since system (7) is ill posed, a null space will exist. Hence, the internal variables cannot be determined uniquely. If, in addition, $n_{\text{tot}} \gg m$ then the danger of overtraining, i.e., memorization, increases considerably, resulting in suboptimal performance. Two reasons cause $\mathbf{A}$ to be rank deficient. First, the sample space may be incomplete, i.e., some samples needed for an accurate optimization are simply missing and some training samples may be erroneously attributed to a wrong class. Second, noise contamination will prevent a perfect fit of both provided and nonprovided data. For example, in a tomographic problem, rank deficiency will already occur if no visited cells are present, making a correct estimate of the true velocities in these cells impossible.

To give an idea of the number of training samples required, the theoretical study of Baum and Haussler (1989) shows that for a desired accuracy level of $(1 - \epsilon)$, at least n_{tot}/ϵ examples must be provided, i.e., $m \geq n_{\text{tot}}/\epsilon$. Thus, to classify 90% of the data correctly, at least 10 times more samples must be provided than internal variables are present, i.e., $m \geq 10 n_{\text{tot}}$.

How can we solve equation (7)? A possible method of estimating the optimal $\mathbf{w}$ is by minimizing the sum of the squared differences between the desired and the actual output of the network. This leads to the least-mean-squares solution, i.e., the weights are determined by solving the normal equations

$$\Delta \mathbf{w} = (\mathbf{A}^t \mathbf{A})^{-1} \mathbf{A}^t \Delta \mathbf{y}, \tag{8}$$

where the superscript (w) is dropped for clarity.

This method, however, has the well-known disadvantage that singularities in $\mathbf{A}^t \mathbf{A}$ cause the divergence of the Euclidean norm $|\Delta \mathbf{w}|$ of the weights, since this norm is inversely proportional to the smallest singular value of $\mathbf{A}$. Moreover, if $\mathbf{A}$ is rank deficient, then this singular value will be zero or at least effectively zero because of a finite machine precision. The squared norm $|\Delta \mathbf{w}|^2$ is also often called the variance of the solution.

To prevent divergence of the solution variance, very often a constrained version of equation (8) is constructed using a positive damping variable β. This method is also known as Levenberg–Marquardt or Tikhonov regularization, i.e., system (8) is replaced by

$$\Delta \mathbf{w} = (\mathbf{A}^t \mathbf{A} + \beta \mathbf{I})^{-1} \mathbf{A}^t \Delta \mathbf{y}, \tag{9}$$

with $\mathbf{I}$ the identity matrix (Lines and Treitel, 1984; Van der Sluis and Van der Vorst, 1987).

The matrix $\mathbf{A}^t \mathbf{A} + \beta \mathbf{I}$ is not rank deficient in contrast to $\mathbf{A}^t \mathbf{A}$. Hence, the solution variance does not diverge but remains constrained. Nevertheless, the method comes at an expense: the solution will be biased because of the regularization parameter β. Therefore, it does not provide the optimal solution in a least-mean-squares sense. The exact value of β must be chosen judiciously to optimize the trade-off between variance and bias (see Van der Sluis and Van der Vorst, 1987; Geman et al., 1992).

More complex regularization can be used on both $\Delta \mathbf{w}$ and $\Delta \mathbf{y}$. For instance, if uncertainty bounds on the output are known (e.g., their variances), then these can be used to rescale the output. A similar rescaling can also be applied on the input and/or weights. This method allows for incorporating any a priori information available. Hence, a complete Bayesian inversion problem can be formulated. See Tarantola (1987) for details on this approach.

Just as in tomography problems, equations (8) and (9) are rarely solved directly. More often an iterative approach is applied. The best known method in neural networks applications is the gradient back-propagation method of Rumelhart et al. (1986) with or without a momentum term, i.e., a term analogous to the function of the regularization factor β. It is a so-called first-order optimization method which approximates $(\mathbf{A}^t\mathbf{A})^{-1}$ in equations (8) and (9) by $\alpha\mathbf{I}$ with $\beta = 0$.

This method is basically a steepest descent algorithm. Hence, all disadvantages of such gradient descent techniques apply. For instance, in the case of curved misfit surfaces, the gradient will not always point to the desired global minimum. Therefore, convergence may be slow (see Lines and Treitel, 1984). To accelerate convergence, the calculated gradients are multiplied with a constant factor α, $0 < \alpha < 2$. However, a judicious choice of α is required, since nonoptimal choices will have exactly the opposite effect, i.e., convergence will slow even further. For instance, if α is too large, strongly oscillating misfits are obtained that do not converge to a minimum; choosing too small a value will slow convergence and possibly hinder the escape from very small local minima. Furthermore, convergence is not guaranteed within a certain number of iterations. In addition, previous ameliorations in the misfit can be partly undone by the next iterations.

Although several improvements have been proposed concerning adaptive modifications of both α (Dahl, 1987; Jacobs, 1988; Riedmiller and Braun, 1993) and more complex regularization terms (Hanson and Pratt, 1989; Weigend et al., 1991; Williams, 1995), the basic algorithm remains identical. Fortunately, other algorithms can be applied to solve the inversion problem. As a matter of fact, any method can be used which solves the normal equations (8) or (9), such as Gauss–Newton methods. Particularly suited are scaled conjugate gradient (SCG) methods, which are proven to converge within $\min(m, n_{\text{tot}})$ iterations, automatically estimate $(\mathbf{A}^t\mathbf{A} + \beta\mathbf{I})^{-1}$ without an explicit calculation, and have a memory of previous search directions, since the present gradient is always conjugate to all previously computed (Møller, 1993; Masters, 1995).

Furthermore, in the case of strongly nonlinear error surfaces with, for example, several local minima, both genetic algorithms and simulated annealing (Goldberg, 1989; Hertz et al., 1991; Masters, 1995) offer interesting alternatives, and hybrid techniques can be considered (Masters, 1995). For instance, simulated annealing can be used to obtain several good initial weight distributions, which can then be optimized by an SCG method. A review of learning algorithms including second-order methods can be found in Battiti (1992). Reed (1993) gives an overview of regularization methods.

A last remark concerning the initialization of the weights. Equation (5) clearly shows the need to start with a good initial guess of these weights. Otherwise, training may become very slow and the risk of falling in local minima increases significantly. Nevertheless, the most commonly used procedure is to apply a random initialization, i.e., $w_i \in [-r, r]$. Even some optimum bounds for r have been established [see, for example, Nguyen and Widrow (1990)].

As mentioned, an alternative procedure is to use a global training scheme first to obtain several good initial guesses to start a localized optimization. However, several theoretical methods have also been developed. The interested reader is referred to the articles of Nguyen and Widrow (1990), who use a linearization by parts of the produced output of the hidden neurons; Denœux and Lengellé (1993), who use prototypes (selected training examples) for an adequate initialization; and Sethi (1990, 1995), who uses decision trees to implement a four-layer neural networks. Another interesting method is given by Karouia et al. (1994) using the theoretical results of Gallinari et al. (1991), who show that a formal equivalence exists between linear neural networks and discriminant or factor analyses. Hence, they initialize their neural networks so that such an analysis is performed and start training from there on.

All of these initialization methods make use of the fact that although linear methods may not be capable of solving all considered applications, they constitute a good starting point for a neural networks. Hence, a linear initialization is better than a random initialization of weights.

Generalization

Now that we are able to train a network, a new question arises: When should training be stopped? It would seem to be a good idea to stop training when a local minimum is attained or when the convergence rate has become very small, i.e., improvement of iteration to iteration is zero or minimal. However, Geman et al. (1992) show that this leads to overtraining, i.e., memorization of the training set: now the noise is fitted, not the global trend. Hence, the obtained weight distribution will be optimal for the training samples, but it will result in bad performance in general. A similar phenomenon occurs in tomography problems, where it is known as overfit (Scales and Snieder, 1998).

Overtraining is caused by the fact that system (7) is ill posed, i.e., a null space exists. The least-mean-squares solution of system (7), equation (8), will result in optimal performance only if a perfect and complete training set is used without any noise contamination. Otherwise, any solution is nonunique because of the existence of this null space. Regularization with equation (9) reduces the influence of the null space but also results in a biased solution, as mentioned earlier.

The classical solution to this dilemma is to use a split set of examples. One part is used for training; the other part is used as a reference set to quantify the general performance (Figure 4). Training is stopped when the misfit of the reference set reaches a minimum. This method is known as holdout crossvalidation.

Although this method generally produces good results, it results in a reduced training set that may pose a problem if only a limited number of examples is available. Because this method

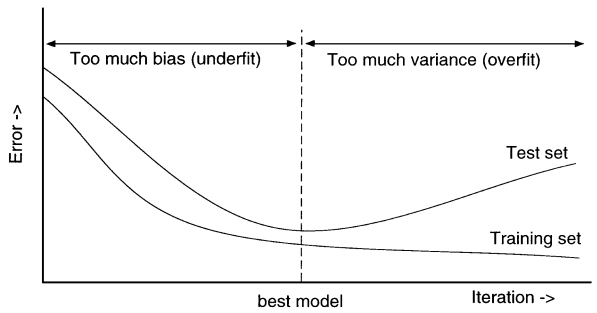

FIG. 4. Generalization versus training error. Adapted from Moody (1994).

requires subdivision of the number of existing examples, the final number of used training samples is reduced even further. Hence, the information contained in the selected examples is not optimally used and the risk of underconstrained training increases.

It is possible to artificially increase the number of training samples m by using noise injection or synthetic modeling to generate noise-free data. However, caution should be used when applying such artificial methods. In the former, small, random perturbations are superimposed on the existing training data. Mathematically, this corresponds to weight regularization (Matsuoka, 1992; Bishop, 1995; Grandvalet and Canu, 1995), thereby only reducing the number of effective weights. Moreover, the noise parameters must be chosen judiciously to optimize again the bias/variance trade-off. In addition, a bad noise model could introduce systematic errors. In the latter case, the underlying model may inadequately represent the real situation, thus discarding or misinterpreting important mechanisms.

To circumvent the problem of split data sets, some other techniques exist: generalized crossvalidation methods, residual analysis, and theoretical measures which examine both obtained output and network complexity.

The problem of holdout crossvalidation is that information contained in some examples is left out of the training process. Hence, this information is partly lost, since it is only used to measure the general performance but not to extract the fundamentals of the considered process. As an alternative, v-fold crossvalidation can be considered (Moody, 1994; Chentouf, 1997).

In this method, the examples are divided into v sets of (roughly) equal size. Training is then done v times on $v-1$ sets, in which each time another set is excluded. The individual misfit is defined as the misfit of the excluded set, whereas the total misfit is defined as the average of the v individual misfits. Training is stopped when the minimum of the total misfit is reached or convergence has become very slow. In the limit of $v=m$, the method is called leave one out. In that case, training is done on $m-1$ examples and each individual misfit is calculated on the excluded example.

The advantage of v-fold crossvalidation is that no examples are ultimately excluded in the learning process. Therefore, all available information contained in the training samples is used. Moreover, training is performed on a large part of the data, namely on $(m-m/v)$ examples. Hence, the optimization problem is more easily kept overconstrained. On the other hand, training is considerably slower because of the repeated crossvalidations. For further details refer to Stone (1974) and Wahba and Wold (1975). Other statistical methods can also be considered, such as the jackknife or the bootstrap (Efron, 1979; Efron and Tibshirani, 1993; Masters, 1995)—two statistical techniques that try to obtain the true underlying statistical distribution from the finite amount of available data without posing a priori assumptions on this distribution. Moody (1994) also describes a method called nonlinear crossvalidation.

Another possible way to avoid a split training set is to minimize theoretical criteria relating the network complexity and misfit to the general performance (Chentouf, 1997). Such criteria are based on certain theoretical considerations that must be satisfied. Some well-known measures are the AIC and BIC criteria of Akaike (1970). Others can be found in Judge et al. (1980). For instance, the BIC criterion is given by

$$\mathrm{BIC} = \ln\left(\frac{\sigma_r^2}{m}\right) + n_{\mathrm{tot}}\frac{\ln m}{m}, \tag{10}$$

where σ_r^2 denotes the variance of the error residuals (misfits). The first term is clearly related to the misfit; the second is related to the network complexity.

These criteria, however, have been developed for linear systems and are not particularly suited to neural networks because of their nonlinear activation functions. Hence, several theoretical criteria had to be developed for such nonlinear systems (MacKay, 1992; Moody, 1992; Murata et al., 1994). Like their predecessors, they are composed of a term related to the misfit and a term describing the complexity of the network. Hence, these criteria also try to minimize both the misfit and the complexity of a network simultaneously.

However, these criteria are extremely powerful if the underlying theoretical assumptions are satisfied and in the limit of an infinite training set, i.e., $m \gg n_{\mathrm{tot}}$. Otherwise they may yield erroneous predictions that can decrease the general performance of the obtained network. Moreover, these criteria can be used only if the neural networks is trained and its structure is adapted simultaneously.

A third method has been proposed in the neural networks literature by Jutten and Chentouf (1995) inspired by statistical optimization methods. It consists of a statistical analysis of the error residuals, i.e., an analysis of the misfit for all output values of all training samples is performed. It states that an optimally trained network has been obtained if the residuals and the noise have the same characteristics. For example, if noise is assumed to be white, training is stopped if the residuals have zero mean and exhibit no correlations (as measured by a statistical test). The method can be extended to compensate for nonwhite noise (Hosseini and Jutten, 1998). The main drawback of this method is that a priori assumptions must be made concerning the characteristics of the noise.

Configuration optimization: Preprocessing and weight regularization

The last remaining problem concerns the construction of a network configuration yielding optimal results. Insight in the way neural networks tackle classification problems already allow for a notion of the required number of hidden layers and the type of neural networks. Nevertheless, in most cases only vague ideas of the needed number of neurons per hidden layer exist.

Classically, this problem is solved by trial and error, i.e., several structures are trained and their performances are examined. Finally, the best configuration is retained. The main problem with this approach is its need for extensive manual labor, which may be very costly, although automatic scripts can be written for construction, training, and performance testing.

In addition, the specific application and its complexity are not the only factors of influence. As shown above, the ratio of the number of total internal variables to the number of training samples is of direct importance to prevent an underconstrained optimization problem. This problem is of immediate concern for applications disposing of large input vectors, i.e., n_i is large, although regularization may help limit the number of effective weights (Hush and Horne, 1993). Very often the number

of required links and nodes can be reduced easily using preprocessing techniques to highlight the important information contained in the input or by using local connections and weight sharing.

Many different preprocessing techniques are available. However, one of the best known is principal component analysis, or the Karhunen–Loève transform. In this approach, training samples are placed as column vectors in a matrix $\mathbf{X}$. The covariance matrix $\mathbf{X}\mathbf{X}^t$ is then decomposed in its eigenvalues and eigenvectors. Finally, training samples and, later, data are projected upon the eigenvectors of the p largest eigenvalues ($p < m$). These eigenvectors span a new set of axes displaying a decreasing order of linear correlation between the training samples. In this way, any abundance in the input may be reduced. Moreover, only similarities are extracted which may reduce noise contamination. The ratio of the sum of the p largest eigenvalues (squared) over the total sum of squared eigenvalues yields an accurate estimate of the information contained in the projected data. More background is provided in Richards (1993) and Van der Baan and Paul (2000).

The matrix $\mathbf{X}$ may contain all training samples, the samples of only a single class, or individual matrices for each existing class. In the latter case, each class has its own network and particular preprocessing of the data. The individual networks are often called expert systems, only able to detect a single class and therefore requiring repeated data processing to extract all classes.

Use of the Karhunen–Loève transform may pose problems if many different classes exist because it will become more difficult to distinguish between classes using their common features. As an alternative, a factor or canonical analysis may be considered. This method separates the covariance matrix of all data samples into two covariance matrices of training samples within classes and between different classes. Next, a projection is searched that simultaneously yields minimum distances within classes and maximum distances between classes. Hence, only a single projection is required. A more detailed description can be found in Richards (1993).

The reason why principal component and factor analyses may increase the performance of neural networks is easy to explain. Gallinari et al. (1991) show that a formal equivalence exists between linear neural networks (i.e., with linear activation functions) and discriminant or factor analyses. Strong indications exist that nonlinear neural networks (such as MLP and RBF networks) are also closely related to discriminant analyses. Hence, the use of a principal component or a factor analysis allows for a simplified network structure, since part of the discrimination and data handling has already been performed. Therefore, local minima are less likely to occur.

Other interesting preprocessing techniques to reduce input can be found in Almeida (1994). All of these are cast in the form of neural networks structures. Notice, however, that nearly always the individual components of the input are scaled to lie within well-defined ranges (e.g., between -1 and 1) to put the dynamic range of the input values within the most sensitive part of the activation functions. This often results in a more optimal use of the input. Hence, it may reduce the number of hidden neurons. For instance, Le Cun et al. (1991) show that correcting each individual input value for the mean and standard deviation of this component in the training set will increase the learning speed. Furthermore, for data displaying a large dynamic range, often the use of $\log(x)$ instead of x is recommended.

Another possible way to limit the number of internal variables is to make a priori assumptions about the neural networks structure and, in particular, about the links between the input and the first hidden layer. For instance, instead of using a fully connected input and hidden layer, only local connections may be allowed for, i.e., it is assumed that only neighboring input components are related. Hence, links between these input nodes and a few hidden neurons will be sufficient. The disadvantage is that this method may force the number of hidden neurons to increase for an adequate description of the problem.

However, if the use of local connections is combined with weight sharing, then a considerable decrease of n_{tot} may be achieved. Thus, grouped input links to a hidden node will have identical weights. Even grouped input links to several nodes may be forced to have identical weights. For large networks, this method may considerably decrease the total number of free internal variables (see Le Cun et al., 1989). Unfortunately, results depend heavily on the exact neural networks structure, and no indications exist for the optimal architecture.

The soft weight-sharing technique of Nowlan and Hinton (1992) constitutes an interesting alternative. In this method it is assumed that weights may be clustered in different groups exhibiting Gaussian distributions. During training, network performance, centers and variances of the Gaussian weight distributions, and their relative occurrences are optimized simultaneously. Since one of the Gaussians is often centered around zero, the method combines weight sharing with Tikhonov regularization. One of the disadvantages of the method is its strong assumption concerning weight distributions. Moreover, no method exists for determining the optimal number of Gaussians, again yielding an architecture problem.

Configuration optimization: Simplification methods

This incessant architecture problem can be solved in two different ways, using either constructive or destructive, i.e., simplification, methods. The first method starts with a small network and simultaneously adds and trains neurons. The second method starts with a large, trained network and progressively removes redundant nodes and links. First, some simplification methods are described. These methods can be divided into two categories: those that remove only links and those that remove whole nodes. All simplification methods are referred to as pruning techniques.

The simplest weight pruning technique is sometimes referred to as magnitude pruning. It consists of removing the smallest present weights and thereafter retraining the network. However, this method is not known to produce excellent results (Le Cun et al., 1990; Hassibi and Stork, 1993) since such weights, though small, may have a considerable influence on the performance of the neural network.

A better method is to quantify the sensitivity of the misfit function to the removal of individual weights. The two best known algorithms proceeding in such a way are optimal brain damage or OBD (Le Cun et al., 1990) and optimal brain surgeon or OBS (Hassibi and Stork, 1993).

Both techniques approximate the variation δE of the least-mean-squares misfit E attributable to removal of a weight w_i

by a second-order Taylor expansion, i.e.,

$$\delta E = \sum_i \frac{\partial E}{\partial w_i}\Delta w_i + \frac{1}{2}\sum_i \frac{\partial^2 E}{\partial w_i^2}(\Delta w_i)^2 + \frac{1}{2}\sum_{i\neq j}\frac{\partial^2 E}{\partial w_i \partial w_j}\Delta w_i \Delta w_j. \quad (11)$$

Higher order terms are assumed to be negligible. Removal of weight w_i implies $\Delta w_i = -w_i$. Since all pruning techniques are only applied after neural networks are trained and a local minimum has been attained, the first term on the right-hand side can be neglected. Moreover, the OBD algorithm assumes that the off-diagonal terms ($i \neq j$) of the Hessian $\partial E^2/\partial w_i \partial w_j$ are zero. Hence, the sensitivity (or saliency) s_i of the misfit function to removal of weight w_i is expressed by

$$s_i = \frac{1}{2}\frac{\partial^2 E}{\partial w_i^2} w_i^2. \quad (12)$$

Weights with the smallest sensitivities are removed, and the neural network is retrained. Retraining must be done after suppressing a single or several weights. The exact expression for the diagonal elements of the Hessian is given by Le Cun et al. (1990).

The OBS technique is an extension of OBD, in which the need for retraining no longer exists. Instead of neglecting the off-diagonal elements, this technique uses the full Hessian matrix $\mathbf{H}$, which is composed of both the second and third terms in the right-hand side in equation (11). Again, suppression of weight w_i yields $\Delta w_i = -w_i$, which is now formulated as $\mathbf{e}_i^t \Delta \mathbf{w} + w_i = 0$, where the vector $\mathbf{e}_i$ represents the ith column of the identity matrix. This leads to a variation δE_i,

$$\delta E_i = \tfrac{1}{2}\Delta \mathbf{w}^t \mathbf{H} \Delta \mathbf{w} + \lambda\left(\mathbf{e}_i^t \Delta \mathbf{w} + w_i\right) \quad (13)$$

(with λ a Lagrange multiplier). Minimizing expression (13) yields

$$\delta E_i = \frac{1}{2}\frac{w_i^2}{H_{ii}^{-1}} \quad (14)$$

and

$$\Delta \mathbf{w} = -\frac{w_i}{H_{ii}^{-1}}\mathbf{H}^{-1}\mathbf{e}_i. \quad (15)$$

The weight w_i resulting in the smallest variation in misfit δE_i in equation (14) is eliminated. Thereafter, equation (15) tells how all the other weights must be adapted to circumvent the need for retraining the network. Yet, after the suppression of several weights, the neural networks is usually retrained to increase performance.

Although the method is well based on mathematical principles, it does have a disadvantage: not only the full Hessian but also its inverse must be calculated. Particularly for large networks, this may require intensive calculations and may even pose memory problems. However, the use of OBS becomes very interesting if the inverse of the Hessian or $(\mathbf{A}^t\mathbf{A} + \beta\mathbf{I})^{-1}$ has already been approximated from the application of a second-order optimization algorithm for network training. The exact expression for the full Hessian matrix can be found in Hassibi and Stork (1993).

Finally, note that equation (11) is only valid for small perturbations Δw_i. Hence, OBD and OBS should not be used to remove very large weights. Moreover, Cottrell et al. (1995) show that both OBD and OBS amount to removal of statistically null weights. Furthermore, their statistical approach can be used to obtain a clear threshold to stop pruning with the OBD and OBS techniques because they propose not to remove weights beyond a student's t threshold, which has clear statistical significance (Hérault and Jutten, 1994).

Instead of pruning only links, whole neurons can be suppressed. Two techniques which proceed in such a way are those of Mozer and Smolensky (1989) and Sietsma and Dow (1991). The skeletonization technique of Mozer and Smolensky (1989) prunes networks in a way similar to OBD and OBS. However, the removal of whole nodes on the misfit is quantified. Again, nodes showing small variations are deleted.

Sietsma and Dow (1991) propose a very simple procedure to prune nodes, yielding excellent results. They analyze the output of nodes in the same layer to detect noncontributing units (NCU). Nodes that produce a near-constant output for all training samples or that have a correlated output to other nodes are removed, since such nodes are not relevant for network performance. A correlated output implies that these nodes always have identical or opposite output. Removal of nodes can be corrected by a simple adjustment of biases and weights of all nodes they connect to. Hence, in principle no retraining is needed, although it is often applied to increase performance. Although Sietsma and Dow (1991) do not formulate their method in statistical terms, a statistical framework can easily be forged and removal can be done on inspection of averages and the covariance matrix. Moreover, this allows for a principal component analysis within each layer to suppress irrelevant nodes.

Both the skeletonization and NCU methods also allow for pruning input nodes. Hence, they can significantly reduce the number of internal variables describing the neural networks, which is of particular interest in the case of limited quantities of training samples.

All pruning techniques increase the generalization capacity of the network because of a decreased number of local minima. Other pruning techniques can be found in Karnin (1990), Pellilo and Fanelli (1993), and Cottrell et al. (1995). A short review of some pruning techniques, including weight regularization methods, is given in Reed (1993).

During pruning, a similar problem occurs as during training: When should pruning be stopped? In practice, pruning is often stopped when the next neural networks cannot attain a predefined maximum misfit. However, this may not be an optimum choice. A better method is to use any of the techniques described in the subsection on generalization. The nonlinear theoretical criteria may be especially interesting because they include the trade-off of network complexity versus misfit.

Configuration optimization: Constructive methods

As a last possibility for creating optimal network structures, we consider the constructive methods. Such methods start from scratch; only input and output layers are defined, and they automatically increase the network size until convergence is reached. The principal problem associated with this approach is to find a suitable stopping criterion. Otherwise, the training set will simply be memorized and generalization will be poor. Mostly theoretical measures evaluating the trade-off between network complexity and performance are used.

Nowadays, constructive algorithms exist for both MLP and RBF networks and even combinations of these, i.e., neural networks using mixed activation functions. Probably the best known constructive algorithm is the cascade correlation method of Fahlman and Lebiere (1990). It starts with a fully connected and trained input and output layer. Next, a hidden node is added which initially is connected only to the input layer. To obtain a maximum decrease of the misfit, the output of the hidden node and the prediction error of the trained network are maximally correlated. Next, the node is linked to the output layer, weights from the input layer to the hidden node are frozen (i.e., no longer updated), and all links to the output layer are optimized. In the next iteration, a new hidden node is added, which is linked to the input layer and the output of all previously added nodes. Again, the absolute covariance of its output and the prediction error of the neural networks is maximized, after which its incoming links are again kept frozen and all links to the output nodes are retrained. This procedure continues until convergence. Each new node forms a new hidden layer. Hence, the algorithm constructs very deep networks in which each node is linked to all others. Moreover, the original algorithm does not use any stopping criterion because input is assumed to be noiseless.

Two proposed techniques that do not have these drawbacks are the incremental algorithms of Moody (1994) and Jutten and Chentouf (1995). These algorithms differ from cascade correlation in that only a single hidden layer is used and all links are updated. The two methods differ in the number of neurons added per iteration [one (Jutten and Chentouf, 1995) or several (Moody, 1994)] and their stopping criteria. Whereas Moody (1994) uses the generalized prediction error criterion of Moody (1992), Jutten and Chentouf (1995) analyze the misfit residuals. Further construction is ended if the characteristics of the measured misfit resemble the assumed noise characteristics.

A variant (Chentouf and Jutten, 1996b) of the Jutten and Chentouf algorithm also allows for the automatic creation of neural networks with several hidden layers. Its general way of proceeding is identical to the original algorithm. However, it evaluates if a new neuron must be placed in an existing hidden layer or if a new layer must be created. Another variant (Chentouf and Jutten, 1996a) allows for incorporating both sigmoidal and Gaussian neurons. It evaluates which type of activation function yields the largest reduction in the misfit (see also Chentouf, 1997).

The dynamic decay adjustment method of Berthold and Diamond (1995) is an incremental method for RBF networks that automatically estimates the number of neurons and the centers and variances of the Gaussian activation functions best providing an accurate classification of the training samples. It uses selected training samples as prototypes. These training samples define the centers of the Gaussian activation functions in the hidden-layer neurons. The weight of each Gaussian represents its relative occurrence, and the variance represents the region of influence. To determine these weights and variances, the method uses both a negative and a positive threshold. The negative threshold forms an upper limit for the output of wrong classes, whereas the positive threshold indicates a minimum value of confidence for correct classes. That is, after training, training samples will at least produce an output exceeding the positive threshold for the correct class and no output of the wrong classes will be larger than the negative threshold.

During training, the algorithm presents the training samples consecutively to the network. If a training sample cannot be correctly classified by the existing network, then this training sample is used as a new prototype. Otherwise, the weight of the nearest prototype is increased to increase its relative occurrence. The variances of all Gaussians describing conflicting classes are reduced such that no conflicting class produces values larger than the negative threshold for this training sample. Output is not bounded because of the linear activation functions which exist in the output nodes. Hence, this is a decision-making network, i.e., it only gives the most likely class for a given training sample but not its exact likelihood.

The dynamic decay adjustment algorithm of Berthold and Diamond (1995) has some resemblance to the probabilistic neural network of Specht (1990). This network creates a Gaussian centered at each training sample. During training, only the optimum, common variance for all Gaussians must be estimated. However, the fact that a hidden node is created for each training sample makes the network more or less a referential memory scheme and will render the use of large training sets very cumbersome. Dynamic decay adjustment, on the other hand, creates new nodes only when necessary.

Other incremental algorithms include orthogonal least squares of Chen et al. (1991), resource allocating network of Platt (1991), and projection pursuit learning of Hwang et al. (1994). A recent review of constructive algorithms can be found in Kwok and Yeung (1997).

PRACTICE

A general strategy

How can these methods and techniques be used in a geophysical application? The following list contains some relevant points to be considered for any application. Particular attention should be paid to the following.

Choice of neural network.—For static problems, a preliminary data analysis or general considerations may already indicate the optimum choice whether to use an MLP or RBF network. For instance, clusters in classification problems are often thought to be localized in input space. Hence, RBF networks may yield better results than MLP networks. However, both types of neural networks are universal approximators, capable of producing identical results. Nevertheless, one type may be better suited for a particular application than the other type because these predictions are asymptotic results. If no indications exist, both must be tried.

Choice of input parameters.—In some problems, this may be a trivial question. In extreme cases, any parameter which can be thought may be included, after which a principal component analysis (PCA) or factor analysis may be used to reduce input space and thereby any redundancy and irrelevant parameters. Nevertheless, an adequate selection of parameters significantly increases performance and quality of final results.

Suitable preprocessing techniques.—Any rescaling, filtering, or other means allowing for an more effective use of the input parameters should be considered. Naturally, PCA or a factor analysis can be included here.

Training set and training samples.—The number of training samples is of direct influence on the total number of internal variables allowed in the neural networks to keep training overconstrained. The total number of internal variables should

never exceed the number of training samples. The largest fully connected neural networks can be calculated using equations (3) and (4). Naturally, such limitations will not exist if a very large training set is available.

Training algorithm and generalization measure.—Naturally, a training algorithm has to be chosen. Conjugate gradient methods yield better performance than the standard backpropagation algorithm since the first is proven to converge within a limited number of iterations but the latter is not. Furthermore, a method has to be chosen to guarantee a good performance in general. This can be any general method—crossvalidation, theoretical measure, or residual analysis. However, these measures must be calculated during training and not after convergence.

Configuration estimation.—The choice between the use of a constructive or a simplification method is important. An increasingly popular choice is using any constructive algorithm to obtain a suitable network configuration and thereafter applying a pruning technique for a minimal optimum configuration. Sometimes, reinitialization and retraining of a neural networks may improve a misfit and allow for continued pruning of the network. In such cases, the reinitialization has allowed for an escape of a local minimum.

An example

To illustrate how some of these methods and techniques can be put in a general methodology, we consider a single example that can be solved relatively easily using a neural networks without the need for complicated processing schemes. Our example concerns the detection and extraction of reflections, ground roll, and other types of noise in a deep seismic reflection experiment to enhance data quality.

Van der Baan and Paul (2000) have shown that the application of Gaussian statistics on local amplitude spectra after the application of a PCA allows for an efficient estimate of the presence of reflections and therefore their extraction. They used a very simple procedure to extract the desired reflections. In a common shot gather (Figure 5a) a particular reflection was picked 14 times on adjacent traces (Figure 5b). Local amplitude spectra were calculated using 128-ms (16 points) windows centered around the picks. These local amplitude spectra

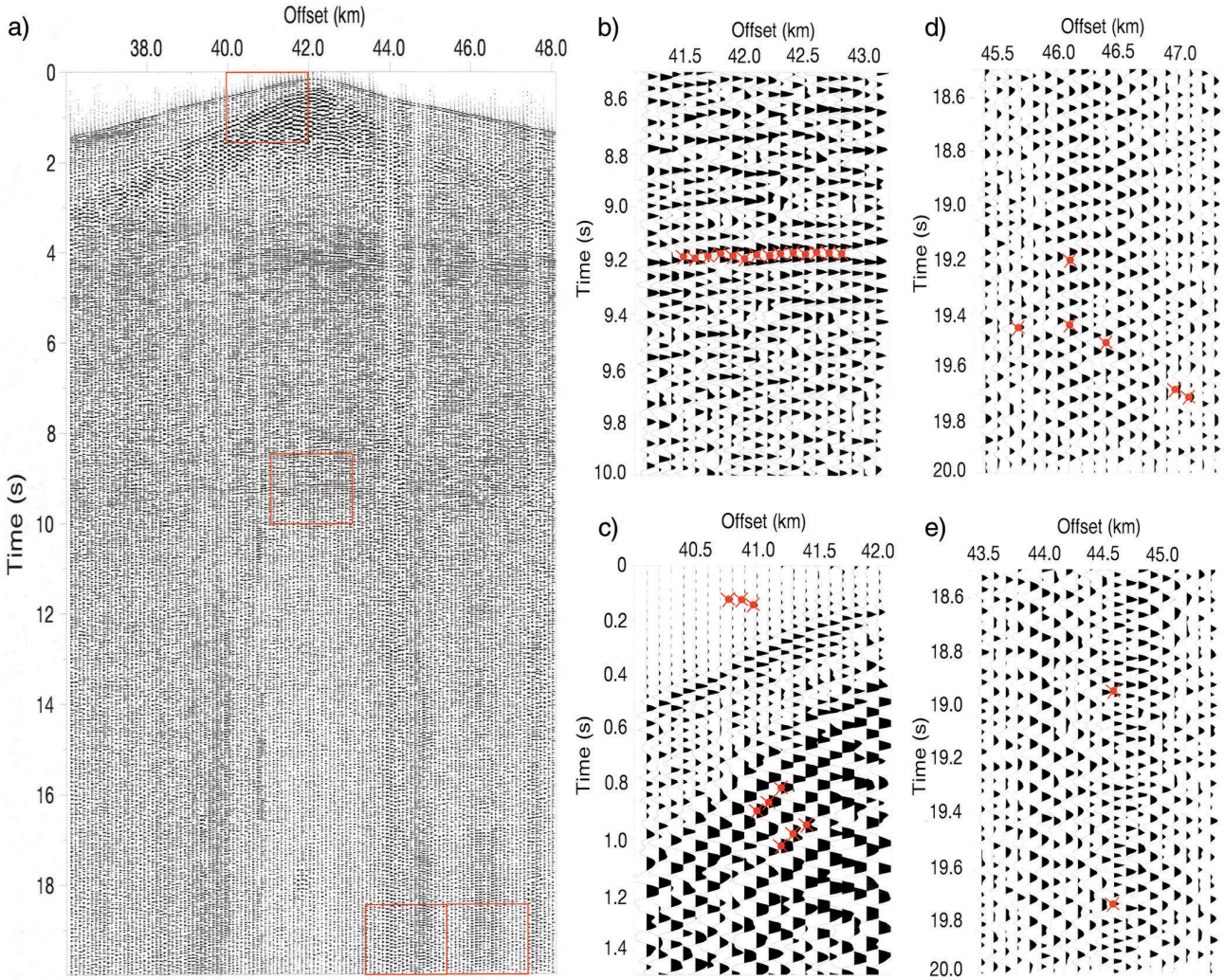

FIG. 5. Common shot gather plus 31 pick positions. (a) Original data, (b) fourteen reflection picks, (c) three prearrival noise plus six ground roll picks, (d) six picks on background noise plus bad traces, and (e) two more picks on bad traces.

were put as column vectors in a matrix **X**, and a PCA was applied using only a single eigenvector. The first eigenvector of $\mathbf{X}\mathbf{X}^t$ was calculated, and henceforth all amplitude spectra were projected upon this vector to obtain a single scalar indicating resemblance to the 14 training samples. Once all 14 amplitude spectra were transformed into scalars, their average and variance were calculated. The presence of reflection energy was then estimated by means of (1) a sliding window to calculate the local amplitude spectra, (2) a projection of this spectrum upon the first eigenvector, and (3) Gaussian statistics described by the scalar mean and variance to determine the likelihood of the presence of a reflection for this particular time and offset. Amplitude spectra were used because it was assumed that a first distinction between signal types could be made on their frequency content. In addition, samples were insensitive to phase perturbations. Extraction results (obtained by means of a multiplication of the likelihood distribution with the original data) are shown in Figure 6a. More details can be found in Van der Baan and Paul (2000).

To obtain a good idea of the possible power of neural networks, we extended their method to detect and extract two other categories of signal: ground roll and all remaining types of noise (including background noise, bad traces, and prearrival noise). To this end, more picks were done on such nonreflections. Hence, ground roll (Figure 5c), background and prearrival noise (Figures 5c and 5d), and bad traces (Figures 5d and 5e) were selected. This resulted in a total training set containing 14 reflections [identical to those used in Van der Baan and Paul (2000)] and 17 nonreflections.

Next, the two other categories of signal were extracted in a similar way as the reflections, i.e., Gaussian statistics were applied to local amplitude spectra after a PCA. Figures 6b and 6c show the results. A comparison of these figures with Figure 5a shows that good results are obtained for the extracted ground roll. However, in Figure 6c many laterally coherent reflection events are visible. Hence, the proposed extraction method did not discern the third category of signals, i.e., all types of noise except ground roll.

Fortunately, the failure to extract all remaining types of noise is easy to explain. Whereas both reflections and ground roll are characterized by a specific frequency spectrum, the remaining types of noise display a large variety in frequency spectra containing both signals with principally only high or low frequencies. Therefore, the remaining types of noise have a multimodal distribution that cannot be handled by a simple Gaussian distribution. To enhance extraction results, the remaining types of noise should be divided into several categories such that no multimodal distributions will exist.

In the following we show how different neural networks are able to produce similar and better results using both MLP and RBF networks. However, we did not want to test the influence of different generalization measures. Hence, results were selected manually—a procedure we do not recommend for general use.

As input parameters, the nine frequencies in the local amplitude spectra are used. These nine frequencies resulted from the use of 16 points (128 ms) in the sliding window. All simulations are performed using the SNNS V.4.1 software package [available from ftp.informatik.uni-stuttgart.de (129.69.211.2)], which is suited for those who do not wish to program their own applications and algorithms.

Equations (3) and (4) indicate that for a training set containing 31 samples and having nine input parameters and three output nodes, already three hidden nodes result in an underconstrained training problem. Two hidden layers are out of the question. Even if expert systems are used (networks capable of recognizing only a single type of signal), then three hidden nodes also result in an underconstrained training problem. On the other hand, expert systems for extracting reflections and ground roll may benefit from PCA data preprocessing because it significantly reduces the number of input parameters and thereby allows for a larger number of hidden neurons.

The first network we used was a so-called 9-5-3 MLP network, i.e., nine input, five hidden, and three output nodes. The network was trained until convergence. The fact that this may have resulted in an overfit is unimportant since the network obtained was pruned using the (NCU) method of Sietsma and Dow (1991). This particular method was chosen because it removes whole nodes at a time (including input nodes), resulting in a 4-2-3 neural networks. The four remaining input nodes contained the second to the fifth frequency component of the amplitude spectra. The resulting signal extractions are displayed in Figure 7. A comparison with the corresponding extraction results of the method of Van der Baan and Paul (2000) shows that more reflection energy has been extracted (Figure 6a versus 7a). Similar results are found for the ground roll (Figure 6b versus 7b). However, the extraction results for the last category containing all remaining types of noise has been greatly improved (compare Figure 6c and 7c). Nevertheless, some laterally coherent energy remains visible in Figure 7c, which may be attributable to undetected reflective energy. Hence, results are amenable to some improvement, e.g., by including lateral information.

The second network to be trained was a 9-5-3 RBF network. Again, the network was trained until convergence and thereafter pruned using NCU. The final network structure consisted of a 5-2-3 neural networks producing slightly worse results than Figure 7c for the remaining noise category, as some ground roll was still visible after extraction. The five remaining input nodes were connected to the first five frequency components of the local amplitude spectra.

Hence, both MLP and RBF networks can solve this particular problem conveniently and efficiently, whereas a more conventional approach encountered problems. In this particular application, results did not benefit from PCA data preprocessing because the distributions were too complicated (mixed and multimodal). However, the use of a factor analysis might have been an option.

Although neither network was able to produce extraction results identical to those obtained by Van der Baan and Paul (2000) for the reflection energy, highly similar results could be obtained using different expert systems with either four input and a single output node or a single input and output node (after PCA preprocessing of data). Thus, similar extraction results could be obtained using very simple expert systems without hidden layers.

DISCUSSION AND CONCLUSIONS

Neural networks are universal approximators. They can obtain an arbitrary close approximation to any continuous function, be it associated with a direct or an inverse problem.

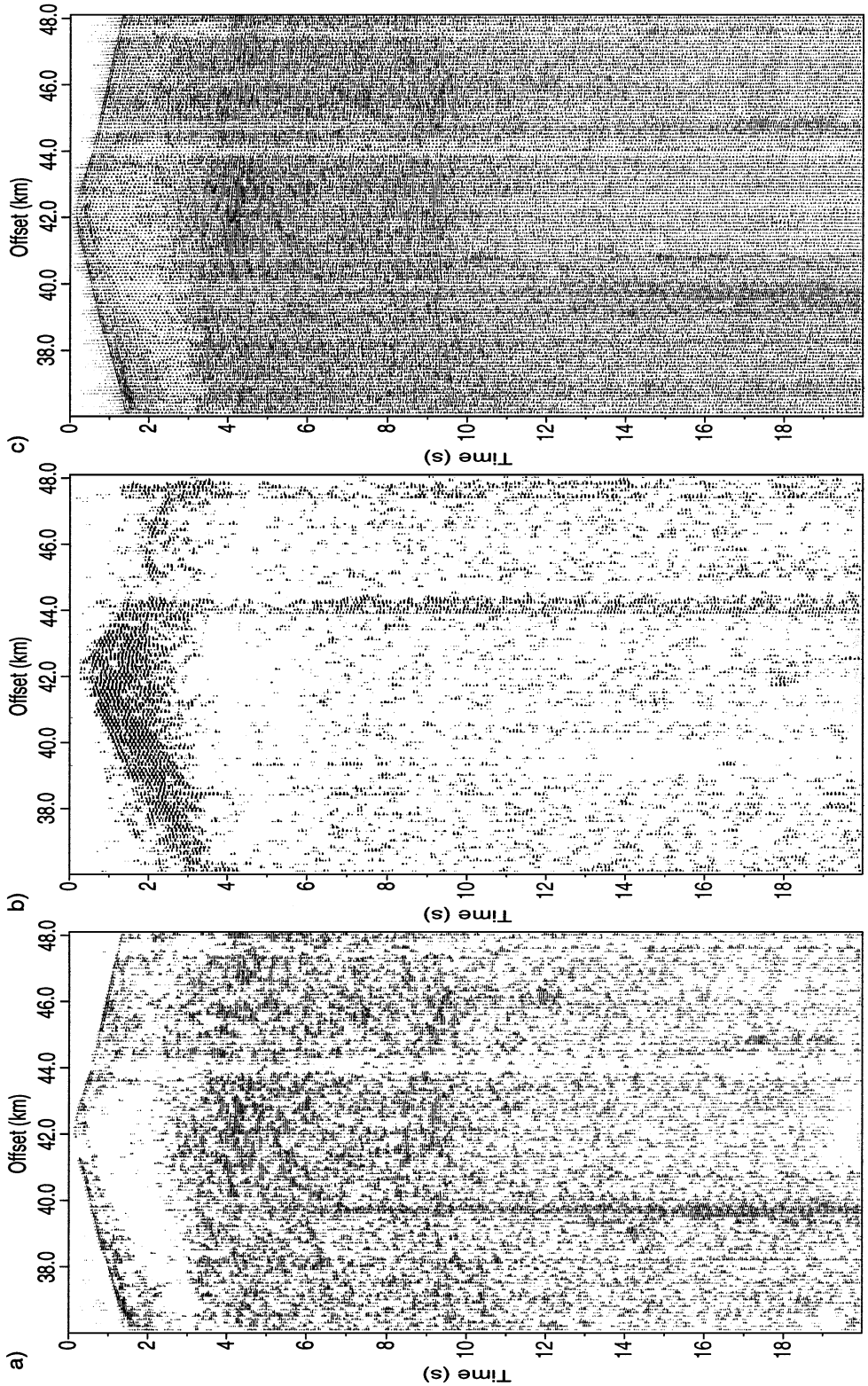

FIG. 6. Extraction results using Gaussian statistics and a PCA for (a) reflections, (b) ground roll, and (c) other types of noise. Compare with Figure 5a.

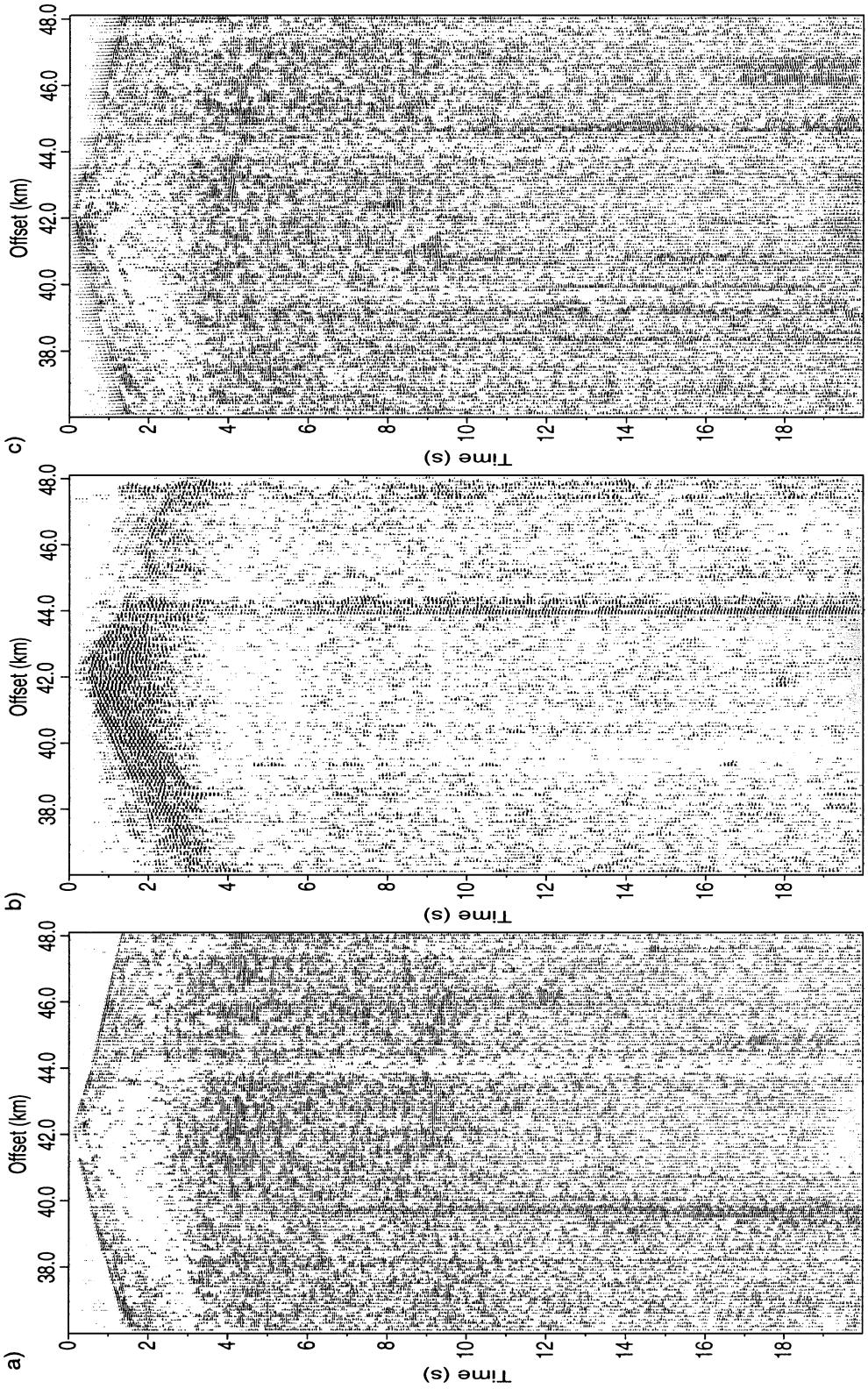

FIG. 7. Extraction results using an MLP network for (a) reflections, (b) ground roll, and (c) other types of noise. Notice the improvement of extraction results for the remaining types of noise.

Therefore, they constitute a powerful tool for the geophysical community to solve problems for which no or only very complicated solutions exist.

We have described many different methods and techniques to facilitate their use and to increase their performance. The last principal issue is related to the training set. As in many other methods, the quality of the obtained results stands or falls with the quality of the training data.

Furthermore, one should first consider whether the use of neural networks for the intended application is worth the expensive research time. Generally, this question reduces to the practical issue of whether enough good training samples can be obtained to guarantee an overconstrained training procedure. This problem may hinder their successful application even after significant preprocessing of the data and reduction of the number of input parameters. If a negative answer must be given to this pertinent question, then a better alternative is the continued development of new and sound mathematical foundations for the particular application.

ACKNOWLEDGMENTS

M. v. d. B thanks Philippe Lesage for an introduction to the domain of neural networks and for pointing to the existence of SNNS. In addition, discussions with Shahram Hosseini are acknowledged. We are grateful for the reviews of Bee Bednar, an anonymous reviewer, and S. A. Levin to whom the note of caution about the use of synthesized data is due.

REFERENCES

Akaike, H., 1970, Statistical predictor identification: Ann. Inst. Statist. Math., **22**, 203–217.

Almeida, L. B., 1994, Neural preprocessing methods, *in* Cherkassy, V., Frieman, J. H., and Wechsler, H., Eds., From statistics to neural networks: Theory and pattern recognition applications: Springer-Verlag, 213–225.

Battiti, R., 1992, First and second order methods for learning between steepest descent and Newton's methods: Neural Comp., **4**, 141–166.

Baum, E. B., and Haussler, D., 1989, What size network gives valid generalization?: Neural Comp., **1**, 151–160.

Berthold, M. R., and Diamond, J., 1995, Boosting the performance of RBF networks with dynamic decay adjustment, *in* Tesauro, G., Touretzky, D. S., and Leen, T. K., Eds., Advances in neural processing information systems **7**: MIT Press, 521–528.

Bishop, C. M., 1995, Training with noise is equivalent to Tikhonov regularization: Neural Comp., **7**, 108–116.

Calderón–Macías, C., Sen, M. K., and Stoffa, P. L., 1997, Hopfield neural networks, and mean field annealing for seismic deconvolution and multiple attenuation: Geophysics, **62**, 992–1002.

——— 1998, Automatic NMO correction and velocity estimation by a feedforward neural network: Geophysics, **63**, 1696–1707.

Carpenter, G. A., and Grossberg, S., 1987, Art2: Self-organization of stable category recognition codes for analog input patterns: Appl. Optics, **26**, 4919–4930.

Chen, S., Cowan, C. F. N., and Grant, P. M., 1991, Orthogonal least squares learning algorithm for radial basis function networks: IEEE Trans. Neural Networks, **2**, 302–309.

Chentouf, R., 1997, Construction de réseaux de neurones multicouches pour l'approximation: Ph.D. thesis, Institut National Polytechnique, Grenoble.

Chentouf, R., and Jutten, C., 1996a, Combining sigmoids and radial basis functions in evolutive neural architectures: Eur. Symp. Artificial Neural Networks, D Facto Publications, 129–134.

——— 1996b, DWINA: Depth and width incremental neural algorithm: Int. Conf. Neural Networks, IEEE, Proceedings, 153–158.

Cottrell, M., Girard, B., Girard, Y., Mangeas, M., and Muller, C., 1995, Neural modeling for time series: A statistical stepwise method for weight elimination: IEEE Trans. Neural Networks, **6**, 1355–1364.

Cybenko, G., 1989, Approximation by superpositions of a sigmoidal function: Math. Control, Signals and Systems, **2**, 303–314.

Dahl, E. D., 1987, Accelerated learning using the generalized delta rule: Int. Conf. Neural Networks, IEEE, Proceedings, **2**, 523–530.

Dai, H., and MacBeth, C., 1994, Split shear-wave analysis using an artificial neural network?: First Break, **12**, 605–613.

Denœux, T., and Lengellé, R., 1993, Initializing back-propagation networks with prototypes: Neural Networks, **6**, 351–363.

Dowla, F. U., Taylor, S. R., and Anderson, R. W., 1990, Seismic discrimination with artificial neural networks: Preliminary results with regional spectral data: Bull. Seis. Soc. Am., **80**, 1346–1373.

Efron, B., 1979, Bootstrap methods: Another look at the Jackknife: Ann. Statist., **7**, 1–26.

Efron, B., and Tibshirani, R. J., 1993, An introduction to the bootstrap: Chapman and Hall.

Fahlman, S. E., and Lebiere, C., 1990, The cascade-correlation learning architecture, *in* Touretzky, D. S., Ed., Advances in neural information processing systems 2: Morgan Kaufmann, 524–532.

Gallinari, P., Thiria, S., Badran, F., and Fogelman–Soulie, F., 1991, On the relations between discriminant analysis and multilayer perceptrons: Neural Networks, **4**, 349–360.

Geman, S., Bienenstock, E., and Doursat, R., 1992, Neural networks and the bias/variance dilemma: Neural Comp., **4**, 1–58.

Goldberg, D. E., 1989, Genetic algorithms in search, optimization and machine learning: Addison-Wesley Publ. Co.

Grandvalet, Y., and Canu, S., 1995, Comments on "Noise injection into inputs in back propagation learning": IEEE Trans. Systems, Man, and Cybernetics, **25**, 678–681.

Hanson, S. J., and Pratt, L. Y., 1989, Comparing biases for minimal network construction with backpropagation, *in* Touretzky, D. S., Ed., Advances in neural information processing systems **1**: Morgan Kaufmann, 177–185.

Hassibi, B., and Stork, D. G., 1993, Second order derivatives for network pruning: Optimal brain surgeon, *in* Hanson, S. J., Cowan, J. D., and Giles, C. L., Eds., Advances in neural information processing systems **5**: Morgan Kaufmann, 164–171.

Hérault, J., and Jutten, C., 1994, Réseaux neuronaux et traitement de signal: Hermès édition, Traitement du signal.

Hertz, J., Krogh, A., and Palmer, R. G., 1991, Introduction to the theory of neural computation: Addison-Wesley Publ. Co.

Hopfield, J. J., 1984, Neurons with graded response have collective computational properties like those of two-state neurons: Proc. Natl. Acad. Sci. USA, **81**, 3088–3092.

Hosseini, S., and Jutten, C., 1998, Simultaneous estimation of signal and noise in constructive neural networks: Proc. Internat. ICSC/IFAC Symp. Neural Computation, 412–417.

Huang, S. C., and Huang, Y. F., 1991, Bounds on the number of hidden neurons in multilayer perceptrons: IEEE Trans. Neur. Networks, **2**, 47–55.

Huang, Z., Shimeld, J., Williamson, M., and Katsube, J., 1996, Permeability prediction with artificial neural network modeling in the Ventura gas field, offshore eastern Canada: Geophysics, **61**, 422–436.

Hush, D. R., and Horne, B. G., 1993, Progress in supervised neural networks—What's new since Lippmann?: IEEE Sign. Process. Mag., **10**, No. 1, 8–39.

Hush, D., Horne, B., and Salas, J. M., 1992, Error surfaces for multilayer perceptrons: IEEE Trans. Systems, Man and Cybernetics, **22**, 1152–1161.

Hwang, J. N., Lat, S. R., Maechler, M., Martin, D., and Schimert, J., 1994, Regression modeling in back-propagation and projection pursuit learning: IEEE Trans. Neural Networks, **5**, 342–353.

Jacobs, R. A., 1988, Increased rates of convergence through learning rate adaptation: Neural Networks, **1**, 295–308.

Judge, G. G., Griffiths, W. E., Hill, R. C., and Lee, T., 1980, The theory and practice of econometrics: John Wiley & Sons, Inc.

Jutten, C., and Chentouf, R., 1995, A new scheme for incremental learning: Neural Proc. Letters, **2**, 1–4.

Karnin, E., 1990, A simple procedure for pruning backpropagation trained neural networks: IEEE Trans. Neural Networks, **1**, 239–242.

Karouia, M., Lengellé, R., and Denœux, T., 1994, Weight initialization in BP networks using discriminant analysis techniques: Neural Networks and Their Applications, Proceedings, 171–180.

Kohonen, T., 1989, Self-organization and associative memory, 3rd ed.: Springer-Verlag New York, Inc.

Kwok, T.-Y., and Yeung, D.-Y., 1997, Constructive algorithms for structure learning in feedforward neural networks for regression problems: IEEE Trans. Neural Networks, **8**, 630–645.

Langer, H., Nunnari, G., and Occhipinti, L., 1996, Estimation of seismic waveform governing parameters with neural networks: J. Geophys. Res., **101**, 20 109–20 118.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., 1989, Backpropagation applied to handwritten zip code recognition: Neural Comp., **1**, 541–551.

Le Cun, Y., Denker, J. S., and Solla, S. A., 1990, Optimal brain damage, *in* Touretzky, D., Ed., Advances in neural information processing systems **2**: Morgan Kaufmann, 598–605.

Le Cun, Y., Kanter, I., and Solla, S., 1991, Eigenvalues of covariance

matrices: Application to neural network learning: Phys. Rev. Lett., **66**, 2396–2399.
Lines, L. R., and Treitel, S., 1984, Tutorial: A review of least-squares inversion and its application to the geophysical domain: Geophys. Prosp., **32**, 159–186.
Lippmann, R. P., 1987, An introduction to computing with neural networks: IEEE ASSP Mag., **4**, No. 2, 4–22.
MacKay, D. J. C., 1992, Bayesian interpolation: Neural Comp., **4**, 415–447.
Masters, T., 1995, Advanced algorithms for neural networks—A C++ sourcebook: John Wiley & Sons, Inc.
Matsuoka, K., 1992, Noise injection into inputs in back-propagation learning: IEEE Trans. Systems, Man, and Cybernetics, **22**, 436–440.
McCormack, M. D., Zaucha, D. E., and Dushek, D. W., 1993, First-break refraction event picking and seismic data trace editing using neural networks: Geophysics, **58**, 67–78.
McCulloch, W. S., and Pitts, W., 1943, A logical calculus of the ideas immanent in nervous activity: Bull. Math. Biophys., **5**, 115–133.
Møller, M. F., 1993, A scaled conjugate gradient algorithm for fast supervised learning: Neural Networks, **6**, 525–533.
Moody, J. E., 1992, The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems, *in* Moody, J. E., Hanson, S. J., and Lippmann, R. P., Eds., Advances in neural information processing systems **4**: Morgan Kaufmann, 847–854.
——— 1994, Prediction risk and architecture selection for neural networks, *in* Cherkassy, V., Frieman, J. H., and Wechsler, H., Eds., From statistics to neural networks: Theory and pattern recognition applications: Springer-Verlag, 213–225.
Mozer, M. C., and Smolensky, P., 1989, Skeletonization: A technique for trimming the fat from a network via relevance assessment, *in* Touretzky, D. S., Ed., Advances in neural information processing systems **1**: Morgan Kaufmann, 107–115.
Murat, M. E., and Rudman, A. J., 1992, Automated first arrival picking: A neural network approach: Geophys. Prosp., **40**, 587–604.
Murata, N., Yoshizawa, S., and Amari, S., 1994, Network information criterion—Determining the number of hidden units for an artificial neural network model: IEEE Trans. Neural Networks, **5**, 865–872.
Nguyen, D., and Widrow, B., 1990, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights: Int. Joint Conf. Neural Networks, Proceedings, **III**, 2063–2068.
Nowlan, S. J., and Hinton, G. E., 1992, Simplifying neural networks using soft weight-sharing: Neural Comp., **4**, 473–493.
Park, J., and Sandberg, I. W., 1991, Universal approximation using radial-basis-function networks: Neural Comp., **3**, 246–257.
Pellilo, M., and Fanelli, A. M., 1993, A method of pruning layered feed forward neural networks, *in* Prieto, A., Ed., International workshop on artificial neural networks: Springer-Verlag, 278–283.
Platt, J., 1991, A resource-allocating network for function interpolation: Neural Comp., **3**, 213–225.
Poulton, M. M., Sternberg, B. K., and Glass, C. E., 1992, Location of subsurface targets in geophysical data using neural networks: Geophysics, **57**, 1534–1544.
Reed, R., 1993, Pruning algorithms—A survey: IEEE Trans. Neural Networks, **4**, 740–747.
Richards, J., 1993, Remote sensing digital image analysis, an introduction: Springer-Verlag New York, Inc.
Riedmiller, M., and Braun, H., 1993, A direct adaptive method for faster backpropagation learning: The RPROP algorithm: Int. Conf. Neural Networks, IEEE, Proceedings, **1**, 586–591.
Romeo, G., 1994, Seismic signals detection and classification using artificial neural networks: Annali di Geofisica, **37**, 343–353.
Röth, G., and Tarantola, A., 1994, Neural networks and inversion of seismic data: J. Geophys. Res., **99**, 6753–6768.
Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986, Learning internal representation by backpropagating errors: Nature, **332**, 533–536.
Scales, J. A., and Snieder, R., 1998, What is noise?: Geophysics, **63**, 1122–1124.
Sethi, I. K., 1990, Entropy networks: From decision trees to neural networks: Proc. IEEE, **78**, 1605–1613.
——— 1995, Neural implementation of tree classifiers: IEEE Trans. Systems, Man and Cybernetics, **25**, 1243–1249.
Sietsma, J., and Dow, R. D. F., 1991, Creating artificial neural networks that generalize: Neural Networks, **4**, 67–79.
Specht, D., 1990, Probabilistic neural network: Neural Networks, **3**, 109–118.
Stone, M., 1974, Cross-validatory choice and assessment of statistical predictions: J. Roy. Statist. Soc., **36**, 111–147.
Tarantola, A., 1987, Inverse problem theory—Methods for data fitting and model parameter estimation: Elsevier Science Publ. Co.
Van der Baan, M., and Paul, A., 2000, Recognition and reconstruction of coherent energy with application to deep seismic reflection data: Geophysics, **65**, 656–667.
Van der Sluis, A., and Van der Vorst, H. A., 1987, Numerical solutions of large, sparse linear algebraic systems arising from tomographic problems, *in* Nolet, G., Ed., Seismic tomography: D. Reidel Publ. Co., 49–83.
Wahba, G., and Wold, S., 1975, A completely automatic French curve: Fitting spline functions by cross-validation: Comm. Stati., **4**, 1–17.
Wang, L.-X., and Mendel, J. M., 1992, Adaptive minimum prediction-error deconvolution and source wavelet estimation using Hopfield neural networks: Geophysics, **57**, 670–679.
Weigend, A. S., Rumelhart, D. E., and Huberman, B. A., 1991, Generalization by weight-elimination with application to forecasting, *in* Lippmann, R. P., Moody, J. E., and Touretzky, D. S., Eds., Advances in neural information processing systems **3**: Morgan Kaufmann, 875–882.
Williams, P. M., 1995, Bayesian regularization and pruning using a Laplace prior: Neural Comp., **7**, 117–143.
Zhang, Y., and Paulson, K. V., 1997, Magnetotelluric inversion using regularized Hopfield neural networks: Geophys. Prosp., **45**, 725–743.