ELSEVIER

Computer Physics Communications 88 (1995) 1–22

Computer Physics Communications

A modified backpropagation algorithm for training neural networks on data with error bars

Klaus A. Gernoth [a], John W. Clark [b]

[a] *International Centre for Theoretical Physics, Strada Costiera 11, 34014 Trieste, Italy*

[b] *McDonnell Center for the Space Sciences and Department of Physics, Washington University, St. Louis, MO 63130, USA*

Received 8 August 1994; revised 15 December 1994

Abstract

A method is proposed for training multilayer feedforward neural networks on data contaminated with noise. Specifically, we consider the case that the artificial neural system is required to learn a physical mapping when the available target values for the output variable are subject to experimental uncertainties, but are characterized by error bars. The proposed method, based on a maximum-likelihood criterion for parameter estimation that allows for nonzero model error, introduces two simple modifications of the on-line backpropagation learning algorithm:

(i) The differential reliability of individual training patterns is taken into account by multiplying the learning-rate constant with a factor $[1 + (\sigma_r/\sigma)^2]^{-1}$, where σ_r is the experimental uncertainty associated with data point r and σ is the current estimate of the model error parameter.

(ii) After each full pass through the training sample, the model error measure σ is updated by solution of a nonlinear algebraic equation, at the current values of the connection weights.

The extended backpropagation algorithm is tested on two problems related to the modeling of atomic-mass systematics by neural networks, the most extensive study being performed for data generated by the liquid-drop mass formula and deliberately obscured by additive Gaussian noise. Provided the underlying mapping is reasonably smooth, neural nets trained with the new procedure are able to learn the true function to a good approximation even in the presence of high levels of noise.

1. Introduction

Artificial neural networks with diverse architectures, dynamics, and learning rules have become popular tools for the solution of a broad range of problems involving optimization, classification, and function approximation [1–6]. The vast majority of applications are driven by engineering and commercial goals. However, recent years have also seen a proliferation of scientific applications, devoted primarily to the statistical analysis, interpretation, and modeling of experimental data. The most visible activity has been centered in chemistry [7,8] and high-energy physics [9–11], but other problem areas have also received attention, including astronomy [12,13], atomic physics [14], and nuclear physics [15–18].

As employed in technology and science, neural networks may be regarded as devices for performing statistical inference [19–31]. They can provide versatile nonparametric computational frameworks for estimation of *a posteriori* probabilities. Under favorable circumstances, these systems may in fact approach the Bayesian ideal [24–27,18].

It is true that a given problem can often be more efficiently attacked with traditional methods [32], especially in those cases where the existing treatments have

SSDI 0010-4655(95)00012-7

been custom-designed and optimized over many years. Even so, the flexibility and relative ease of implementation of neural-network techniques have proven attractive to many researchers.

In the present paper we shall focus on the use of neural networks to perform a nonlinear regression analysis for a given set of data consisting of pairs of sample inputs and corresponding target responses. In practice, the regression is carried out by supervised training of a multilayer feedforward network [1–6]: a cost function that measures the overall error of network response is minimized with respect to the connection-weight parameters characterizing the network model. Backpropagation [1], based on a mean-square cost function and an incremental gradient-descent minimization technique, is by far the most commonly used training algorithm.

In dealing with experimental data, or data produced by stochastic computer simulation, one must confront the question of how experimental or statistical errors will affect the learning process and in turn the model that results from the training procedure. It is certainly undesirable for the network to learn (and hence fit) spurious features created by such errors. On the other hand, one would like to make some use of the information that is contained in noisy data points. Here we shall propose and test a simple extension of the standard backpropagation algorithm that may permit accurate modeling to be achieved even when the output variables – and thus the nominal targets for learning – are afflicted with noise. The new algorithm, which derives from the well-known maximum-likelihood criterion [33] in the version delineated by Möller and Nix [34], is predicated on a prior assignment of "error bars" to the data points r that have been furnished for analysis. To be definite, we assume that the experimental (or stochastic-computational) errors are normally distributed with zero mean and known standard deviations σ_r, although the approach is a general one and can be adapted to other characterizations of the noise.

The new learning algorithm is developed in Section 2. (Those interested only in the statement of the modified backpropagation routine may skip directly to Section 2.4.) In Section 3, the behavior of the proposed algorithm is explored within the specific context of the modeling of atomic masses. In the principal example, relatively small multilayer feedforward nets are asked to approximate the semiclassical liquid-drop mass formula in the presence of noise that is elevated to high levels.

It is not the purpose of this paper to engage the issue of whether neural networks are superior or inferior to traditional methods. Rather, our goal is limited and relative: we seek to demonstrate the superiority of the modified backpropagation procedure compared to the original version that takes the noisy training data at face value. Nevertheless, some comments on the larger issue will be made in Section 4, and an update of the competitive status of neural-network modeling of the atomic-mass table will be provided in Section 3.3.

The proposed extension of backpropagation learning is expected to prove valuable not only within the scientific context of experimental data analysis, but also in numerous technological applications where the training sample is subject to noise.

2. Formulation of the modified training procedure

Our objective is to adapt the backpropagation learning algorithm to a situation in which the modeler is given training data that contain experimental errors but is also provided with an uncertainty estimate for each datum. The formal steps leading to this objective are the following. We first cast the modeling problem in general terms and make certain assumptions about the statistics of the data and of the model that is to describe it. We then invoke the maximum-likelihood criterion (MLC) as a basis for estimating the parameters that complete the model specification. The usual application of the MLC that yields the familiar least-squares fitting prescription [35] is nontrivially extended to include the estimation of a model error measure along with the intrinsic model parameters that are adjusted to fit the data [34]. Specializing to the case of a multilayer feedforward neural network, the latter parameters are identified with the weights of the network connections. We then generalize the on-line backpropagation procedure to allow iterative solution of the coupled optimization problems implied by the MLC.

2.1. Problem statement: modeling in the presence of noise

The mapping that underlies the phenomenon being modeled is denoted $\mathbf{x} \rightarrow \mathbf{Y}$, where $\mathbf{Y}$ is the set of dependent quantities subject to measurement and $\mathbf{x}$ is the set of "independent" variables that determine $\mathbf{Y}$ through some natural law or through some corresponding "true" theory. The problem at hand is to reconstruct this mapping – or rather to construct a model that approximates it – based on a restricted set of instances of the map, these instances being corrupted by noise. It will be convenient to refer to $\mathbf{x}$ as the input and $\mathbf{Y}$ as the output. For simplicity, we suppose that there is only one dependent variable Y. The data made available for the modeling process (i.e., the training data) consist of n pairs $(\mathbf{x}^{(r)}, y_e^{(r)})$, where $y_e^{(r)}$ is the experimentally determined value of Y corresponding to the input value $\mathbf{x}^{(r)}$. In addition, for each of the data points r an experimental uncertainty $\sigma_e^{(r)}$ in the Y measurement has been specified. As is common practice in statistical modeling, some of the data (with associated uncertainty assignments) is set aside for the purpose of testing the predictive power of the resulting model on examples that were not used in the determination of the fitting parameters. A model is not considered acceptable unless it shows respectable performance in generalizing to novel examples.

Three key assumptions are made concerning the statistical properties of the model and of the experimental measurement:

(I) The model mapping $\mathbf{x} \rightarrow y_m$ differs from the true mapping $\mathbf{x} \rightarrow Y$ in a large number of small terms with fluctuating signs and magnitudes, such that the central limit theorem applies [34]. More especially, for the model outputs corresponding to the training sample we have

$$y_m^{(r)} = Y^{(r)} + \epsilon_m^{(r)} , \quad r = 1, \ldots, n , \tag{1}$$

where the discrepancy $\epsilon_m^{(r)} \in N(0, \sigma_m^2)$ from the true value $Y^{(r)}$ is a normally distributed random variable with zero mean and standard deviation σ_m.

(II) The experimental values supplied for Y also contain stochastic errors that are normally distributed with zero mean, but with instance-specific standard deviations that are identified with the assigned uncertainties, i.e.,

$$y_e^{(r)} = Y^{(r)} + \epsilon_e^{(r)} \tag{2}$$

with $\epsilon_e^{(r)} \in N\big(0, (\sigma_e^{(r)})^2\big)$.

(III) The input variables of the problem, $\mathbf{x}$, are not subject to noise.

2.2. Maximum-likelihood estimation of model standard deviation and model parameters

Our purposes are best served by shifting from what would appear to be the natural view and regarding the *true* value $Y^{(r)}$ in (1), rather than the *model* output $y_m^{(r)}$, as the random variable. This accords with the fact that the model, by construction, will give a definite output value for each input $\mathbf{x}^{(r)}$. Möller and Nix [34] offer a vivid parable to illustrate the operational situation: God uses the phenomenological model fashioned by His human creations, but with intrinsic parameters He decides (and which we must estimate); to the resulting $y_m^{(r)}$ He adds a random number $-\epsilon_m^{(r)} \in N(0, \sigma_m^2)$ to achieve a result $Y^{(r)}$ to His liking.

Adopting this viewpoint, and noting from assumptions (I) and (II) that

$$y_e^{(r)} = Y^{(r)} + \epsilon_e^{(r)} = y_m^{(r)} + \epsilon_m^{(r)} + \epsilon_e^{(r)} , \tag{3}$$

we may form the conditioned probability density for obtaining the value ξ for the random variable $y_e^{(r)}$ as

$$f_{y_e^{(r)}}\big(\xi | y_m^{(r)}, (\sigma_e^{(r)})^2 + \sigma_m^2\big) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{(\sigma_e^{(r)})^2 + \sigma_m^2}} \times \exp\left[-\frac{(\xi - y_m^{(r)})^2}{2\big((\sigma_e^{(r)})^2 + \sigma_m^2\big)} \right] . \tag{4}$$

In this expression, $y_m^{(r)}$ is to be treated as a parameter of the distribution, along with the quantity σ_m, which is a kind of "sanitized" root-mean-square (rms) model error. The model output $y_m^{(r)}$ is in turn determined by the intrinsic parameters ω_p, $p = 1, \ldots, P$, that specify the particular model being applied (which is ideally the one selected and used by God, within the Möller-Nix parable). The parameter σ_m will ultimately be used as a sensible measure of the quality of fit achieved by the model with respect to the true values $Y^{(r)}$ of Y, rather than the noisy values $y_e^{(r)}$.

At this stage, the problem has been reduced to a familiar one of estimating the unknown parameters (namely, the model error measure σ_m and the fitting parameters ω_p) of a known statistical distribution (namely, Eq. (4)). To perform the estimations, we appeal to the maximum-likelihood criterion (MLC) [33–36,19]. For n data points r, the likelihood function L is formed as the product of n densities $f_{y_e^{(r)}}\left(\xi^{(r)}|\sigma_m^2,\{\omega_p\}\right)$ of the type (4), where we now display only the essential conditioning parameters. Maximizing $\log L$ with respect to the model parameters $\{\omega_p\}$ and σ_m, we have $P+1$ equations to be solved for these $P+1$ quantities:

$$\sum_{r=1}^{n} \frac{y_e^{(r)} - y_m^{(r)}}{(\sigma_e^{(r)})^2 + \sigma_m^2} \frac{\partial y_m^{(r)}}{\partial \omega_p} = 0 \quad (i = 1,\ldots,P)\ , \tag{5}$$

$$\sum_{r=1}^{n} \frac{(y_e^{(r)} - y_m^{(r)})^2 - [(\sigma_e^{(r)})^2 + \sigma_m^2]}{[(\sigma_e^{(r)})^2 + \sigma_m^2]^2} = 0\ . \tag{6}$$

In Bayesian terms, this optimization strategy effectively assumes uniform priors for the parameters being determined. Technically, the MLC provides a consistent estimate, which, however, may be biased [33]. Bias, which introduces a fractional error in an estimated parameter of order P/n, may be reduced by taking a larger data sample, if that option is open.

As is readily seen, solution of (5) and (6) for the model error measure σ_m and the intrinsic model parameters ω_p is equivalent to minimizing the function

$$\frac{1}{2}\tilde{\chi}^2 = \sum_{r=1}^{n} \frac{(y_e^{(r)} - y_m^{(r)})^2}{2[(\sigma_e^{(r)})^2 + \sigma_m^2]} \tag{7}$$

with respect to the ω_p (while holding σ_m fixed at the value determined by Eq. (6)) and solving Eq. (6) for σ_m (the $y_m^{(r)}$ that enter being evaluated at the minimizing set of ω_p). Contact with the ordinary least-squares procedure for determining optimal fitting parameters is made by setting $\sigma_m \equiv 0$ and dropping Eq. (6). The least-squares criterion is based on the working assumption that the model, with appropriately determined intrinsic parameters, is the truth; hence the model standard deviation σ_m makes no appearance in the probability density (4) or in the likelihood function L.

2.3. Maximum-likelihood estimation for a feedforward network model

We next specialize the general prescription afforded by Eqs. (5)–(6) to statistical modeling with feedforward neural networks. In particular, the experimental value $y_e^{(r)}$ of property Y corresponding to the example or "pattern" r (or more generally a coded version of $y_e^{(r)}$) is identified with the target activity value $t^{(r)}$ for a single output unit that delivers the result of the network's computation of Y. Similarly, the model value $y_m^{(r)}$ is identified with the actual output of this unit, $a^{(r)}$, upon presentation of pattern r, i.e., when the input-layer units are activated so as to encode $\mathbf{x}^{(r)}$. Further streamlining the notation, we henceforth abbreviate the experimental uncertainty $\sigma_e^{(r)}$ as σ_r and the model error measure σ_m as σ. The adjustable parameters ω_p of the network model are of course the weights of the connections and the biases of the neuronal units, generically denoted $w_{ll'}$. (As is customary, we regard the bias w_{lo} of unit l as the weight of a connection from a fictitious neuron o with perpetual activity $a_o \equiv 1$.)

Assuming that $\sigma_m = \sigma$ remains finite (surely the case in any nontrivial application), multiplication of Eq. (7) by σ_m^2 yields the alternative cost function

$$E = \sum_{r=1}^{n} E_r \tag{8}$$

wherein

$$E_r = \frac{1}{2}\frac{(t^{(r)} - a^{(r)})^2}{1 + (\sigma_r/\sigma)^2} \tag{9}$$

is a pattern-specific error. Minimization of this objective function with respect to the intrinsic model parameters $\{\omega_p\} \equiv \{w_{ll'}\}$, for fixed σ, is again equivalent to solution of Eq. (5) for given σ_m. The choice of (8)–(9) as cost function rather than (7) leads to a more natural generalization of the conventional backpropagation algorithm (colloquially referred to as "vanilla" backprop). Indeed, if we set all the experimental uncertainties σ_r to zero, E defined by (8)–(9) becomes just the mean-square cost function $E^{[o]}$ of the usual treatment. As with vanilla backprop, the nominal goal of the modified backpropagation scheme is to minimize the pattern-summed cost function, a goal approached by stepwise adjustment of the weights $w_{ll'}$

in accordance with error signals propagated backward from the output layer [1,3]. A new feature is that as the minimization progresses, the unknown model-error parameter σ in the denominator of E_r is to be re-determined consistently by solution of Eq. (6). Converting to neural-network notation, it is advantageous to recast this equation in a form suitable for iterative solution [34]:

$$\sigma^2 = \sum_{r=1}^{n} c_r \left[(t^{(r)} - a^{(r)})^2 - \sigma_r^2 \right] , \tag{10}$$

where c_r is defined by

$$c_r = \left(\sigma_r^2 + \sigma^2\right)^{-2} \Big/ \sum_{s=1}^{n} \left(\sigma_s^2 + \sigma^2\right)^{-2} . \tag{11}$$

One may start the iteration of (10) with the simple rms measure of model error

$$\sigma^{[0]} = \left[\frac{1}{n} \sum_{r=1}^{n} (t^{(r)} - a^{(r)})^2 \right]^{1/2} , \tag{12}$$

commonly adopted in the noise-free problem. Eq. (10) gives concrete expression to the notion that the experimental errors should somehow be "subtracted out" in estimating the quality of the model, which is now measured by σ.

Again as in ordinary backprop, the weight parameters $w_{ll'}$ enter the objective function explicitly through the nested functional dependence of the output activities $a^{(r)}$. However, the cost function E of (8)–(9) also carries a further *implicit* dependence on the weights through the parameter σ, since Eq. (10) also involves the $a^{(r)}$.

2.4. The noise-adapted backpropagation algorithm ("cleanprop")

In imitation of standard backprop, the noise-adapted learning algorithm that we propose and test is based on a gradient-descent approach to minimization of the cost function in weight space. As is common practice [1,3], we depart from strict gradient descent by updating weights after each pattern presentation rather than after each full pass through the training sample (i.e. after each "epoch"). Thus we implement an "on-line" (or "stochastic") updating scheme, which has certain advantages [3,37] over the "batch" (or "epochal") procedure, notably an enhanced possibility of escape from local minima of E.

The modified backpropagation algorithm, or *cleanprop*, consists of the following steps.

(i) Initialize the set of weight parameters $w_{ll'}$ at a randomly selected point in weight space.

(ii) Present a randomly selected input pattern to the network, corresponding to the rth exemplar, and calculate the response $a^{(r)}$ of the output unit.

(iii) *Holding the model error parameter σ of Eq. (10) at its current value* (see step (v)), update the adjustable weights via

$$\Delta w_{ll'}^{(r)} = -\eta \frac{\partial E_r}{\partial w_{ll'}} , \tag{13}$$

where E_r is the cost (9) associated with pattern r and η is a constant learning rate.

(iv) Repeat (ii)–(iii) until the system has been exposed (without repetition) to all the patterns, thus completing an epoch.

(v) *Holding the weights at their current values*, solve Eq. (10) for σ by iteration, with the ordinary rms model error measure $\sigma^{[0]}$ of Eq. (12), or the latest value of σ, as input (convergence usually requires only a few iterations).

(vi) Repeat (ii)–(v) for as many epochs as needed to achieve convergence of the learning process, with neither the model error parameter σ nor the cost function E of Eq. (8) showing significant further decrease.

The resulting "asymptotic" value of σ is a natural measure of the accuracy with which the network fits the training sample while taking account of the uncertainties associated with the data points, i.e., it is a measure of the intrinsic learning performance. Further evidence on the degree to which the network model has captured the underlying map $\mathbf{x} \to Y$ is provided by the corresponding error measure σ for predictive performance. The latter is obtained by solving the nonlinear Eq. (10) for the case that the sum on r runs over the *test* set, with the weights taken at their values for the mature network.

2.5. Behavior of the cleanprop algorithm – consequences of pattern-specific learning

The essence of the proposed learning algorithm – which gives less influence to those training patterns with larger error bars – can be easily understood in terms of a pattern-specific learning rate. We can write the weight-update rule (13) in the same form as in ordinary on-line backprop, namely as

$$\Delta w_{ll'}^{(r)} = -\tilde{\eta}\,\frac{\partial E_r^{[o]}}{\partial w_{ll'}}\,, \tag{14}$$

by introducing a modified or "effective" learning rate

$$\tilde{\eta} = \eta \Big/ \left[1+\left(\frac{\sigma_r}{\sigma}\right)^2\right], \tag{15}$$

where η is the "bare" learning-rate parameter fixed at a suitable value $0 < \eta < 1$ and $E_r^{[o]}$ is the cost of pattern r in vanilla backprop (namely E_r of Eq. (9) with $\sigma_r = 0$).

Referring to Eqs. (14) and (15), several interpretive comments on the proposed learning procedure may now be offered.

(a) If the standard deviation σ_r assigned to the data point r is large compared to the current value of the model error measure σ, then the change in weights is small. In other words, experimental data with large error bars have little weight in the learning (fitting) process – which is just the qualitative behavior we seek.

(b) Developing further on this behavior, assume that, in the course of the training procedure, σ has become very small compared to all the σ_r (as when a very good model has been created). Then the effective learning rate $\tilde{\eta}$ will approach zero, and significant weight changes cease to occur. Thus, the new algorithm has the salutary property that training fades out as the model improves and the estimated error σ becomes negligible. In the limiting situation where σ is nearly zero, this inherent weight-change decay prevents a "near-perfect" neural-network model from being destroyed in further training cycles. By contrast, in vanilla backprop the quality of a network model may deteriorate with prolonged training as the system learns the spurious features of the noisy data.

(c) In the opposite case where the experimental errors are all negligible compared to σ (as in the extreme example of a bad model of the results of good experiments, or perhaps in the early phases of training), the learning algorithm reduces to vanilla backprop, since $\tilde{\eta}$ becomes identical with η in Eq. (15). This reduction is foretold at an earlier point in the development: with $\sigma_r << \sigma$, $r = 1, \ldots, n$, the usual rms expression (12) for σ may be retrieved from Eq. (6).

(d) The basic assumption (I) may fail in the initial stages of learning, when systematic errors (attributable to the special choice of starting weights) may dominate. But with large model errors, the effect of experimental errors can be ignored and the procedure is equivalent to standard backprop.

(e) What if the experimental standard deviation is the same for all patterns? Then the denominator on the right in Eq. (15) is just some pattern-independent factor that adjusts the learning rate $\tilde{\eta}$ depending on the value of σ for the current epoch. Within the limitations of its parametric resources, the net will learn the given data indiscriminately, errors and all, as in standard backprop. However, if η is held fixed during the learning process, it is still true that weight modifications are quenched as the model improves.

In connection with observations (b) and (e), it may be noted that staged reductions in learning rate have proven beneficial to the creation of high-quality models in problem contexts where the data are regarded as noise-free (for example, see [16,17]).

2.6. Generalizations of the proposed learning algorithm

As is sometimes indicated by the nature of the problem [38], we may relax the assumption (I) that the model values for the quantity Y differ from those generated with the true mapping by an additive normally distributed random number with *zero* mean; instead we may take $\epsilon_m^{(r)} \in N(\mu_m, \sigma_m^2)$ in Eq. (1). Eqs. (4)–(7) are generalized to arbitrary mean error μ_m by the replacement $y_m^{(r)} \to y_m^{(r)} + \mu_m$. In the new version of (5), the differentiation with respect to the fitting parameters ω_p is to be performed with *both* σ_m and μ_m held constant, and Eq. (6), which determines the model standard deviation σ_m, is to be supplemented by a second equation

$$\sum_{r=1}^{n} \frac{y_e^{(r)} - y_m^{(r)} - \mu_m}{(\sigma_e^{(r)})^2 + \sigma_m^2} = 0 \,, \tag{16}$$

determining the mean model error μ_m. Corresponding replacements are to be made in the modified backpropagation algorithm. Thus, writing $\mu_m = \mu$, the pattern-specific cost E_r defined by Eq. (9) and appearing in Eq. (13) becomes

$$E_r = \frac{1}{2} \frac{(t^{(r)} - a^{(r)} - \mu)^2}{1 + (\sigma_r/\sigma)^2} \,, \tag{17}$$

and, after each epoch, the estimates of both σ and μ are to be updated via self-consistent iterative solution of

$$\sigma^2 = \sum_{r=1}^{n} c_r \left[(t^{(r)} - a^{(r)} - \mu)^2 - \sigma_r^2 \right] \tag{18}$$

and

$$\mu = \sum_{r=1}^{n} c_r (t^{(r)} - a^{(r)}) \,, \tag{19}$$

with the c_r again given by (11). It should be clear that adaptation of the scheme to other types of model/experimental statistics, deviating from the orthodox Gaussian cases, is also possible.

Extension of the analysis and procedure to the case of several dependent or output variables Y_k, $k = 1, \ldots, K$, each a function of $\mathbf{x}$, is straightforward. It is also worth pointing out explicitly that the raw material provided for the modeling process – the noisy data and attendant uncertainty estimates – need not be of experimental origin, but might instead be generated by computer simulation, especially by a stochastic algorithm like Monte Carlo.

3. Numerical applications

We now describe two applications of the modified backpropagation procedure introduced in Section 2. The results will illustrate the effectiveness of the new cleanprop learning scheme in suppressing the effects of "experimental" errors when training neural networks on noisy data.

3.1. Application 1: liquid-drop model of atomic masses

The first application, although rather academic in nature, is the more informative as a test of the proposed method, since it allows comparison of the output $y_m^{(r)}$ of the neural-network model with the corresponding *true* value $Y^{(r)}$ of the property Y. The true values of the output quantity are generally unknown in real-world applications, where they are usually obscured by noise or experimental error. To examine instead a situation that is fully under our control, we choose some well-defined function $Y(\mathbf{x})$ and generate a collection of pairings $(\mathbf{x}^{(r)}, Y^{(r)})$ for a list of inputs $\mathbf{x}^{(r)}$. These pairings constitute the true data. The true values $Y^{(r)}$ are then artificially contaminated with noise in accordance with basic assumption (II), the "experimental" error $\epsilon_e^{(r)} = \epsilon_r$ for each instance r being randomly drawn from a Gaussian distribution with zero mean and prescribed standard deviation (uncertainty) σ_r. From the resulting set of noisy examples, $(\mathbf{x}^{(r)}, y_e^{(r)} = Y^{(r)} + \epsilon_r)$, a certain fraction, selected at random, furnish nominal targets $t^{(r)} = y_e^{(r)}$ for the learning process. The remaining examples form the test set. By construction, the test and training samples have the same statistical properties, provided both are sufficiently large.

Such a diagnostic application is more illuminating if the problem addressed is not purely academic. Accordingly, we have chosen a "true" function $Y(\mathbf{x})$ that corresponds to a schematic version of a real-world scientific problem that has been previously studied with neural-network techniques [15,16,39]. Multilayer feedforward nets have been taught to model the systematics of the atomic-mass table with some precision (see Section 3.3). In this case, the components of the input quantity $\mathbf{x}$ are the proton number Z and the neutron number N of a given nuclide, or nucleus, while the dependent variable Y is the mass excess

$$\Delta m(Z, N) = m(Z, N) - A m_u \tag{20}$$

of that nuclide. The mass excess is the difference between the mass $m(Z, N)$ of the neutral atom and a reference mass given by the number of nucleons in the nucleus, $A = N + Z$, times the atomic mass unit $m_u = 931.48$ MeV, which is one-twelfth the mass of a neutral ^{12}C atom.

In the earlier work, the neural-network systems were trained on masses that were (mostly) determined by experiment and thus include complex effects associated with deformation, like-nucleon pairing, and nuclear shell structure, along with a host of more subtle aspects of real nuclei. Here we consider the much simpler problem (called the liquid-drop problem) in which the "true" masses are generated not by nature but instead by the familiar Bethe-Weizsäcker semi-empirical mass formula [40]

$$m(Z,N) = Zm_H + Nm_n - a_1 A + a_2 A^{2/3} + a_3 Z^2/A^{1/3} + a_4 (Z-N)^2/A \,, \qquad (21)$$

wherein m_H and m_n are the masses of the hydrogen atom and the neutron. The last four terms in (21) are motivated by a liquid-drop model of the nucleus and represent, in order, the volume, surface, Coulomb, and (a)symmetry contributions to the mass. We adopt the coefficients a_1 = 15.68 MeV, a_2 = 18.56 MeV, a_3 = 0.717 MeV, and a_4 = 28.1 MeV determined empirically by Myers and Swiatecki [41]. For our initial exploration of the cleanprop algorithm, the "true" function $Y(\mathbf{x})$ is identified with the mass-excess function of the liquid-drop model, given by Eq. (20) with $m(Z,N)$ calculated from Eq. (21). It should be noted that we disregard the pairing and shell corrections that are sometimes attached to the liquid-drop formula.

To define the test problem completely we must also specify the standard deviations σ_r associated with the data points $(\mathbf{x}^{(r)}, y_e^{(r)} = t^{(r)})$. Considering the physical interpretation of the function $Y(\mathbf{x})$ selected, it might seem most reasonable to assume a distribution of σ_r values such that that larger error bars are assigned to nuclides in the (N,Z) plane lying farther from the line of stable nuclei. However, for our purposes it is clearly more instructive to assume a generic distribution rather than one that is so context-specific. Thus we have assigned uncertainties σ_r to the individual "measurements" r by sampling a Gaussian distribution with zero mean and standard deviation Γ, i.e., $\sigma_r \in N(0,\Gamma^2)$. The behavior of the cleanprop algorithm has been studied as Γ is increased in steps to larger and larger values and the noise tends to swamp the signal.

The computational study proceeds with "true" mass-excess values $Y^{(r)} = \Delta m(Z,N)$ derived from formulas (20)–(21) – and associated "experimental" errors ϵ_r – for 2251 nuclides r. These nuclei, which belong to the set of 2291 used in most of the earlier neural-network studies of the atomic-mass table [15,16,39,42], lie in or relatively near the valley of stability and span most of the domain of stable and unstable nuclides observed in nature. All pairs of learning/test runs for the liquid-drop problem are based on the same training and test sets. The training set is composed of 1700 randomly chosen nuclides, leaving 551 examples for testing predictive ability. Fig. 1 shows the locations, in the (N,Z) plane, of the nuclides in these training and test samples.

The neural networks that are taught to model the systematics of the liquid-drop masses have the standard layered, feedforward architecture in which all forward connections between nodes in successive layers are present, no backward or lateral connections are permitted, and connections do not skip layers. The input layer contains two units, whose activities encode the values of Z and N, while the output layer consists of a single unit displaying a coded value for the corresponding mass excess produced by the network. In addition to the input and output layers, the networks possess two or three intervening hidden layers. For the transfer (or "activation") function g that maps the stimulus $u_l = \sum_{l'} w_{ll'} a_{l'}$ felt by a hidden or output neuron l into its activity a_l, we adopt the conventional logistic sigmoid

$$a_l = g(u_l) = \frac{1}{e^{-u_l} + 1} \,, \qquad (22)$$

which implies activities in the range [0,1].

Analog representations are employed at input and output layers. An appropriate shift and scaling factor are used to transform the target mass excess (including "experimental" error) to a target activity on [0,1]; the inverse transformation converts the actual activity of the output unit to a mass excess in MeV. Similarly, scaling factors of 1/120 and 1/140, respectively, are applied to fit the relevant Z and N values into the interval [0,1]. (It is not strictly necessary to scale or otherwise transform the inputs to the network since the input units merely function as inert registers for values of the quantity $\mathbf{x}$. However, it has been argued [43] that such preprocessing is desirable.)

We shall not be concerned with achieving precision fits of the mapping $Y(\mathbf{x})$ defined by the liquid-drop model, since our primary aim is to establish the utility

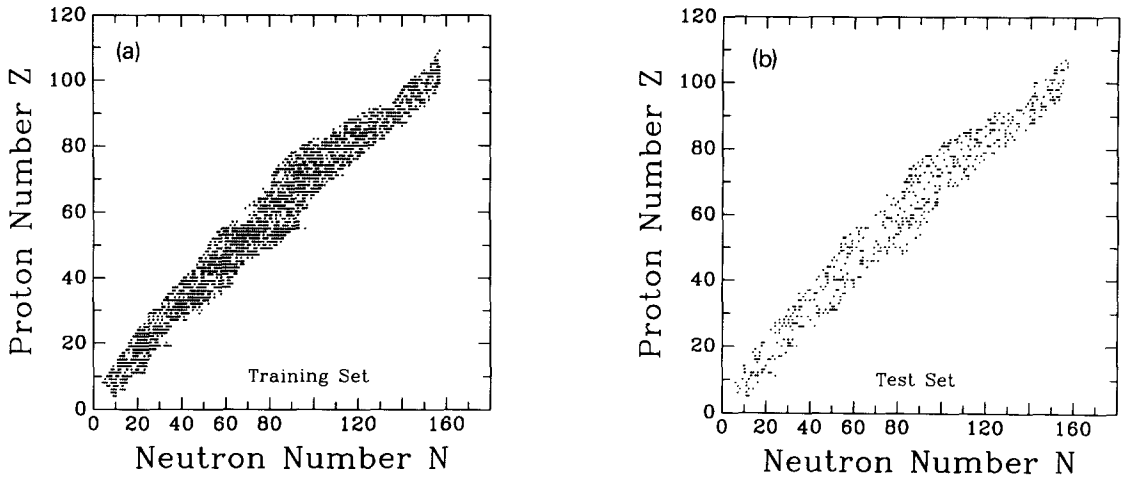

Fig. 1. Training nuclides (a) and test nuclides (b) for the problem of fitting the liquid-drop mass formula in the presence of noise.

of the new learning procedure in reducing the effect of target noise in supervised training of neural networks. Nevertheless, it is well to demonstrate that highly accurate modeling of the liquid-drop data set is possible in the noise-free case. The results of some training runs based on standard backprop are collected in Table 1. Here, and throughout, we make use of the notation for network types introduced in Ref. [15]. For example, $^{a}(2+5+5+5+1)_{a}[81]$ is the designation for an exclusively feedforward network having 2 input units, three successive hidden layers of 5 units each, and one output unit, with analog coding schemes at both input and output interfaces and 81 modifiable weight/bias parameters.

Inspecting the rms-error entries in Table 1, we see that all eight of the sample models perform essentially as well on the test set as they do on the training set. This behavior is often taken as an indication of good generalization, although the more conservative statement is that all the features learned by such networks are common features of training and test samples. Clearly, some benchmark is needed to judge the accuracy of these models. The scale of the rms error estimates $\sigma^{[o]}$ appearing in Table 1 can be compared with the mass-excess values themselves, which range from about -90 MeV up to some 170 MeV. Even the simplest of the four network types considered, $^{a}(2+3+3+3+1)_{a}[37]$, is capable of reproducing both the test and the training data with a typical relative error below 3%. Another point of reference is offered by conventional mass modeling, wherein traditional nuclear models and theories are used to construct fits of the experimental mass excesses. A fit having an rms error in the range 0.2–0.3 MeV would be regarded as competitive, even with $\sim$ 500 adjustable parameters [44,45]. Most popular among the conventional fits of the empirical data are the macroscopic-microscopic models of Möller and Nix [34,38,46] (favored because of their high theoretical input and small number of free parameters). The latest versions have σ values of 0.669 MeV for the finite-range droplet model (with 38 input parameters) and 0.779 MeV for the finite-range liquid-drop model (36 input parameters) [46]. It is true that the realistic masses present a more difficult problem than the liquid-drop problem considered here, since the physical mass surface $m(Z, N)$ has much more structure than the smooth function (21). On the other hand, significant improvements on the performance shown in Table 1 are undoubtedly available as needed, since it is possible to fit any Borel-measurable map from a finite-dimensional input space to a finite-dimensional output space with arbitrary accuracy using some three-layer feedforward network [47]. Some specific practical avenues to enhanced performance in learning and/or prediction are indicated in Section 3.3, in a status report on the problem of accurate modeling of the experimental mass data.

It is to be emphasized, however, that high-precision

Table 1
Performance of neural network models on the liquid-drop mass problem for noiseless data

Net type	N(epochs)	$\sigma^{[o]}$(train)	$\sigma^{[o]}$(test)
$^a(2+3+3+3+1)_a$[37]	20000	1.336	1.216
	20000	3.942	3.992
	20000	4.436	4.335
$^a(2+5+5+5+1)_a$[81]	3000	0.518	0.602
	10000	0.295	0.301
	20000	0.245	0.232
$^a(2+7+7+1)_a$[85]	5000	0.630	0.622
$^a(2+9+9+1)_a$[127]	3000	0.602	0.580

Performance in learning and prediction is measured by the conventional rms error $\sigma^{[o]}$ of Eq. (12), quoted in MeV. The data consists of mass excess values given by Eqs. (20)–(21) for a set of 2251 nuclides, which is divided by random selection into a training set of 1700 nuclei and a complementary test sample of 551 cases. The number of epochs of training is denoted N(epochs). The three entries for the first network architecture correspond to different sets of initial weights, whereas the same set of initial weights is used in all runs for the second architecture.

regression analysis is *not* the point of the present exercise. Rather, our intention is to demonstrate the superiority of cleanprop *relative to* vanilla backprop when dealing with noisy data. Hence it is clearly not advisable to introduce extraneous complications (unusual cost functions, connections that skip layers, pruning recipes, etc.), which might cloud the comparison. Likewise, it would not be prudent to seek higher accuracy of fit through a substantial increase in the number of hidden units. Such a recourse might well be damaging to predictive behavior, unless the number of training examples is commensurately increased – but this option is not open if we wish to simulate the conditions of a real-world problem. All these considerations suggest that the baseline backpropagation networks listed in Table 1 should be well suited to our purposes.

We now turn to the specifics of actual implementation of the modified backpropagation procedure. The new procedure is in fact extremely convenient to apply since, between σ updates via Eq. (10), it operates just like standard backprop except for the presence of a pattern-specific learning rate (15). Calculating the derivative in Eq. (13) (or Eq. (14)), the prescription for on-line updating of weights (and biases) upon presentation of pattern r takes the more explicit form

$$\Delta w_{ll'}^{(r)} = \eta \left[1 + \left(\frac{\sigma_r}{\sigma}\right)^2\right]^{-1} \delta_l^{(r)} a_{l'}^{(r)} + \alpha \Delta w_{ll'}^{(r-1)} \tag{23}$$

in terms of error signals $\delta_l^{(r)}$, where l is a hidden or output unit and l' is a hidden or input unit. Here we have supplemented the rule (13) with a momentum term [1,3] to suppress oscillations in weight space that may occur during learning. The index $r-1$ refers to the pattern presented immediately prior to the current pattern r. The momentum parameter has been set to $\alpha = 0.9$ in all numerical applications. The error signals $\delta_l^{(r)}$ appearing in the update rule (23) are given by exactly the same expressions as in ordinary backpropagation [1,3].

In the runs to be described, the initial weights and biases are drawn from a uniform distribution in the interval $[-0.5, 0.5]$. For each epoch of training, the order of presentation of patterns is refreshed by random permutation. Thus the training process follows a stochastic path in pattern space. In conformity with the training scheme (i)–(vi) specified in Section 2.4, the quality figure σ is readjusted after each epoch. Like the rms error (12) in ordinary stochastic backprop, the error measure σ determined by Eq. (10) exhibits oscillations when plotted against the number of completed epochs. Performance is reported for networks corresponding to those weights found to yield the minimal error σ over the pre-set number of learning epochs. (The nets of Table 1 were obtained using the analogous selection criterion within ordinary backprop, based on minimal $\sigma^{[o]}$.)

We have chosen the $^a(2+3+3+3+1)_a$[37]

Table 2
Test of the modified backpropagation procedure ("cleanprop") for training multilayer feedforward nets on noisy data

Run	Rule	Train			Test		
		$\sigma^{[o]}$	σ	$\sigma^{[true]}$	$\sigma^{[o]}$	σ	$\sigma^{[true]}$
1(5)1	cp	6.615	4.990	5.074	6.196	4.072	4.436
•(0)1	bp	5.031	5.031	5.031	4.415	4.415	4.415
1(5)2	cp	4.762	1.847	2.037	4.609	1.582	1.891
1(5)2	bp	4.641	1.780	1.885	4.608	1.988	1.995
•(0)2	bp	2.285	2.285	2.285	2.105	2.105	2.105
1(5)3	cp	6.316	4.575	4.709	6.430	4.521	4.645
1(5)3	bp	7.084	5.573	5.728	7.148	5.428	5.433
•(0)3	bp	1.617	1.617	1.617	1.642	1.642	1.642
1(5)4	cp	4.622	1.399	1.610	4.491	1.693	1.797
•(0)4	bp	1.244	1.244	1.244	1.305	1.305	1.305
2(5)1	cp	6.115	4.159	4.387	5.927	3.648	4.158
2(5)2	cp	4.754	1.895	2.109	4.622	1.773	2.016
2(5)3	cp	6.285	4.560	4.758	6.267	4.730	4.690
2(5)4	cp	4.607	1.292	1.568	4.433	1.484	1.715
3(5)1	cp	4.609	1.581	1.768	4.376	1.612	1.766
3(5)2	cp	4.746	1.934	2.096	4.391	1.754	1.963
3(5)3	cp	4.766	1.795	2.002	4.460	1.735	1.853
3(5)4	cp	4.506	1.430	1.574	4.369	1.676	1.757

The quoted results refer to networks of type $^{a}(2+3+3+3+1)_{a}[37]$. For all training runs, the prescribed maximum number of epochs was 20000, the learning rate $\eta = 0.5$, and the momentum parameter $\alpha = 0.9$. The quantity $\sigma^{[o]}$ is the root-mean-square error calculated on the basis of the noisy data, $\sigma = \sigma_m$ is the model error, and $\sigma^{[true]}$ is the root-mean-square error with respect to the true, noise-free mass-excess values provided by the liquid-drop model; all three measures are quoted in MeV. The training set consists of 1700 and the test set of 551 patterns. The "experimental" error ϵ_r contained in the target output value for pattern r is chosen at random from a pool of 400 values normally distributed with zero mean and standard deviation σ_r. The experimental standard deviation for pattern r was itself chosen randomly from a normal distribution with zero mean and halfwidth $\mathcal{W} = 5$ MeV. The notation $i_e(\mathcal{W})i_i$ is used to distinguish the different paired training/test runs. The index i_e designates a specific choice of experimental errors ϵ_r, while i_i labels a specific choice of initial weights and stochastic training path in pattern space. Noise-free training/test runs are denoted $\bullet(0)i_i$. The training rule implemented is either cleanprop (cp) or ordinary backprop (bp).

network type for thorough study. This model type admits satisfactory accuracy in both learning and prediction, while allowing a large number of training experiments to be carried out in a relatively short time. A representative collection of results from systematic cleanprop runs on noise-corrupted liquid-drop data is displayed in Tables 2–4, for four different values of $\mathcal{W} = \Gamma\sqrt{2\ln 2}$, the halfwidth at half maximum of the normal distribution $N(0, \Gamma^{-2})$ that is sampled to produce the "experimental" uncertainties σ_r. The quality of individual networks, in learning and prediction, is assessed in terms of three performance figures, $\sigma^{[o]}$, σ, and $\sigma^{[true]}$. The first two are the usual rms error measure defined by Eq. (12) and the theoretical error estimate calculated from Eq. (10). Since, in the present application, the actual values of the desired quantity – the mass excess in the liquid-drop model – are known, we are also in a position to form the rms deviation of the mass excesses produced by the model from their respective *true* (as opposed to "experimental") values,

$$\sigma^{[true]} = \left[\frac{1}{n}\sum_{r=1}^{n}(Y^{(r)} - y_m^{(r)})^2\right]^{1/2}. \qquad (24)$$

This is the third performance measure appearing in the tables. It furnishes the most telling indication of how well a network is modeling the actual function.

Different pairs of training/test entries for a given noise level $\mathcal{W}$ belong to different realizations of (a) the set of random choices of the "experimental" errors ϵ_r from the distributions $N(0, \sigma_r^2)$ and/or (b) the starting point in weight space and the stochastic training path in pattern space. All entries for a given $\mathcal{W}$

Table 3
Test of the cleanprop procedure for training multilayer feedforward nets on noisy data

Run	Rule	Train			Test		
		$\sigma^{[o]}$	σ	$\sigma^{[true]}$	$\sigma^{[o]}$	σ	$\sigma^{[true]}$
1(10)1	cp	9.094	2.267	2.752	8.644	2.470	2.828
1(10)1	bp	11.200	6.159	6.417	10.938	6.274	6.280
•(0)1	bp	1.186	1.186	1.186	1.157	1.157	1.157
1(10)2	cp	9.916	4.435	4.756	8.994	3.745	4.355
•(0)2	bp	3.158	3.158	3.158	2.782	2.782	2.782
1(10)3	cp	8.955	1.461	1.903	8.404	1.718	1.874
1(10)3	bp	1.708	1.708	1.708	1.869	1.869	1.869
1(10)4	cp	9.120	2.315	2.685	8.552	2.206	2.742
•(0)4	bp	2.564	2.564	2.564	2.529	2.529	2.529
2(10)1	cp	8.532	1.366	1.960	8.513	1.706	2.086
2(10)1	bp	11.012	6.188	6.385	10.822	5.752	6.128
2(10)2	cp	9.798	4.942	5.081	9.712	4.093	4.427
2(10)3	cp	8.432	0.949	1.482	8.429	1.122	1.324
2(10)4	cp	8.660	2.048	2.733	8.681	2.671	2.688
3(10)1	cp	8.840	1.613	2.159	8.272	1.920	2.307
3(10)1	bp	11.389	6.239	6.416	10.731	5.726	6.183
3(10)2	cp	10.166	4.845	5.002	9.276	3.962	4.401
3(10)3	cp	8.815	1.566	1.915	8.356	1.655	1.850
3(10)4	cp	8.918	2.372	2.705	8.663	2.678	2.730

As in Table 2, except that the experimental standard deviations σ_r are chosen from a Gaussian distribution with halfwidth $\mathcal{W} = 10$ MeV. Again the same set of σ_r is used in all runs.

correspond to the same choice of σ_r values. To provide baselines for judging the quality of the network models developed using the cleanprop algorithm, Tables 2–4 include some reference entries for nets trained by ordinary backpropagation, either on the noise-free data or on standardized noisy data. Each of these backprop nets shares the same starting point/stochastic path and (if applicable) the same noisy data with one of the cleanprop nets. (It should be noted that combinations of starting points/stochastic paths bearing the same label (e.g. $i_i = 1, 2, 3,$ or 4) are not in general identical. In Table 4, combinations with the same label do coincide but the output coding schemes applied at the two noise levels $\mathcal{W} = 15$ and 25 MeV differ somewhat.)

A number of pertinent conclusions may be drawn from these tables.

- The performance measure $\sigma^{[o]}$ is typically comparable to (or somewhat larger than) the standard deviation $\Gamma \approx 0.85\,\mathcal{W}$ of the normal distribution from which the uncertainties σ_r were chosen. This is expected, since the networks in question have far too few adjustable parameters to fit the given data (37 weights and biases to reproduce data generated by 1700 random variables). With a substantial increase in the number of processing units, a feedforward network model could precisely fit the noisy training sample – but this is *not* desirable, since predictive performance would most probably suffer.
- That cleanprop is effective in screening out the noise is documented by the fact that σ and $\sigma^{[true]}$ are closely comparable in size and are in most cases considerably smaller than both $\sigma^{[o]}$ and Γ. The latter feature persists over the range of noise levels $\mathcal{W}$ involved, even out to $\mathcal{W} = 25$ MeV, which corresponds to very heavy contamination. The only overt sign of deterioration of the cleanprop models as the noise amplitude increases is a growing discrepancy between σ and $\sigma^{[true]}$ (evident in Table 4).
- At the larger noise levels, the cleanprop networks have much better performance figures than nets trained by backprop on the same data. Even so, ordinary backpropagation does show some ability to filter out noise. This ability may be attributed – in part – to the smoothness of the semi-empirical mass formula.
- Examining the results for $\sigma^{[true]}$, it is seen that some

Table 4
Test of the cleanprop procedure for training multilayer feedforward nets on noisy data

Run	Rule	Train			Test		
		$\sigma^{[o]}$	σ	$\sigma^{[true]}$	$\sigma^{[o]}$	σ	$\sigma^{[true]}$
1(15)1	cp	13.263	0.717	1.504	12.178	1.021	1.447
1(15)1	bp	14.138	5.762	6.117	13.495	5.753	5.936
1(15)2	cp	13.460	2.678	3.308	12.578	3.517	3.366
1(15)2	bp	14.389	6.177	6.675	13.836	6.496	6.480
1(15)3	cp	13.497	2.972	3.603	12.606	2.958	3.799
2(15)1	cp	13.620	0.901	1.901	11.939	1.030	2.115
2(15)2	cp	13.864	2.381	3.122	11.865	3.322	3.135
2(15)3	cp	14.253	3.925	4.224	12.309	4.674	4.507
3(15)1	cp	12.957	0.470	1.284	12.198	1.056	1.109
3(15)2	cp	13.197	2.530	3.215	12.396	3.198	3.236
3(15)3	cp	12.995	0.737	1.475	12.281	1.064	1.650
1(25)1	cp	22.051	1.027	1.921	20.241	0.860	1.927
1(25)2	cp	22.193	2.232	3.217	20.498	2.463	3.409
1(25)3	cp	22.026	1.126	1.867	20.184	2.018	1.875

As in Table 2, except that the experimental standard deviations σ_r are chosen from a Gaussian distribution with halfwidth $\mathcal{W}$ = 15 or 25 MeV. Again the same set of σ_r is used in all runs for given $\mathcal{W}$.

of the models obtained via cleanprop are reasonably good on an absolute scale, even when the data used to train them are extremely noisy. Cases in point are provided by the 1(15)1, 3(15)1, 3(15)3, 1(25)1, and 1(25)3 cp entries in Table 4. Moreover, there are examples in the tables to demonstrate that cleanprop is capable of processing corrupted data to yield network models with $\sigma^{[true]}$ values comparable to those of networks trained with backprop on noise-free data, using the same starting weights and training path in pattern space.

- The quality of models depends sensitively on the starting point in weight space and/or the stochastic path in pattern space. For given $\mathcal{W}$, the quality of solution is generally not very sensitive to the specific choice of "experimental" errors ϵ_r, although there are some examples represented in the tables for which this choice does noticeably alter the learning trajectory in weight space. (In Table 2, compare the 3(5)1 entry with the 1(5)1 and 2(5)1 entries and the 3(5)3 entry with the 1(5)3 and 2(5)3 entries; and, in Table 4, the 3(15)3 case with the 1(15)3 and 2(15)3 results.) Variations of model performance on shifting from one noise level to another are quite visible. However, as attested by further results not shown, such variations are typically rather mild if the starting point and stochastic pattern path are left unchanged. The "experimental" errors should of course exert a greater influence on the final network configuration as $\mathcal{W}$ is increased, assuming that the learning process is not quenched.
- Predictive accuracy relative to the true function (as measured by $\sigma^{[true]}$) is comparable to learning accuracy. This is a good sign that the network is seeing through the noise and making a respectable fit of the underlying function. Satisfactory generalization is facilitated by the smoothness of the function to be learned, which is believed to be responsible for the fact that in many cases $\sigma^{[o]}$ is slightly smaller for the test set than for the training set.

The basic conclusion is that the cleanprop algorithm works very well in this application. Even when the data are contaminated with large errors, the modified backpropagation scheme can yield good approximations to the true mapping. A comparative study of the behavior of the method at extreme values of $\mathcal{W}$, namely 25, 50, and 75 MeV, indicates that failure of the algorithm with increasing noise level is extraordinarily graceful. (This study examines sets of cases in which the initial weights and pattern path are kept the same, while fixed distributions of the σ_r and the ϵ_r are appropriately scaled with $\mathcal{W}$.) To some extent, such

robustness must depend on the very smooth character of the target mapping.

Whatever the problem, the cleanprop procedure must ultimately cease to be useful in the high-noise regime, if for no other reason than the following. If all σ_r become very large, the effective learning rate will become very small even for moderately large σ. Thus the learning process will stagnate before a good model has been achieved.

As a testing ground for the new learning algorithm, the liquid-drop application elaborated above has two advantages over most real-world problems: (i) a known mathematical relationship between the input variables and the desired dependent variable, and (ii) a known mechanism for masking this dependent variable with known "experimental" uncertainties. Naturally, the latter uncertainties are generated in accordance with key assumption (II) of Section 2.1. On the other hand, the validity of key assumption (I) is still subject to question in the present application, since it depends on unknown aspects of the neural-network modeling process. To establish the consistency of the demonstration, we have made a number of explicit checks of the validity of assumption (I). For nets trained with cleanprop on the noise-corrupted liquid-drop data, the distribution of model errors $\epsilon_m^{(r)}$ is found to conform approximately with the corresponding Gaussian $N(0, \sigma_m^2 = \sigma^2)$. An example is shown in Fig. 2.

3.2. Application 2: modeling experimental mass systematics

Continuing the theme of mass modeling, it is natural to perform an analysis of *actual* experimental nuclidic mass data. As a database for this exercise we employ 1654 mass-excess values $y_e^{(r)} = \Delta m_e^{(r)}(Z^{(r)}, N^{(r)})$ together with their assigned uncertainties $\sigma_e^{(r)}$, provided to us by Möller and Nix. Evidently, in this problem we no longer have the advantage of knowing the true values $Y^{(r)}$ of the mass excess. On the other hand, the data now reflect the discontinuous character of the realistic mass surface $m(Z, N)$, which is attributable primarily to pairing and shell effects, with smaller apparent irregularities yet to be explained in physical terms.

The cleanprop learning algorithm has been seen to function remarkably well in filtering out random errors superimposed on the semi-empirical mass formula (21). However, it is expected to be less effective when the underlying function to be approximated is jagged or erratic instead of smooth. Our second example serves to test the procedure on a problem that contains far more complex intrinsic structure, although the test is not a very stringent one since the experimental error bars are generally quite small in this case. (The average uncertainty, over the full data set, is only $\bar{\sigma}_r = 0.0412$ MeV. The largest uncertainties, reaching 1.5 MeV, occur on the boundaries of the domain of observed nuclei.)

For the purposes of our study, the given database is divided into a training set of 1323 nuclides, selected at random, and the complementary test set of 331 examples. In choices of architecture and coding, we follow the lead of earlier statistical modeling of the experimental atomic-mass table in terms of feedforward neural networks [15,16,39]. Accordingly, we consider networks of the type $^h(18+10+10+10+1)_a[421]$, which, on implementing various refinements of standard backpropagation training, has been shown to admit precise modeling of the raw data (see Section 3.3). The input coding scheme utilizes two sets of eight "on-off" neurons representing Z and N in binary, plus two analog neurons that (redundantly) encode scaled values of $A = N + Z$ and $N - Z$. Specifications are otherwise essentially the same as in the simpler problem of capturing the liquid-drop formula, and application of the cleanprop training rule proceeds in a similar manner.

Results obtained with the actual data are presented in Table 5. The rms error measure $\sigma^{[o]}$ and the derived theoretical error figure σ, calculated for training and test sets, are quoted for networks developed from two different starting points in weight space through different paths in pattern space. Since the experimental mass data are of high quality, the improvement gained by cleanprop is expected to be marginal, and the difference $\sigma^{[o]} - \sigma$ is indeed found to be very small. In passing, we note that good generalization ability is also exhibited by these more realistic mass models.

3.3. Neural-network mass models: comparison with conventional methods

The problem of fitting and prediction of atomic masses – with data supplied either by a physical model and artificially obscured with noise, or by experiment

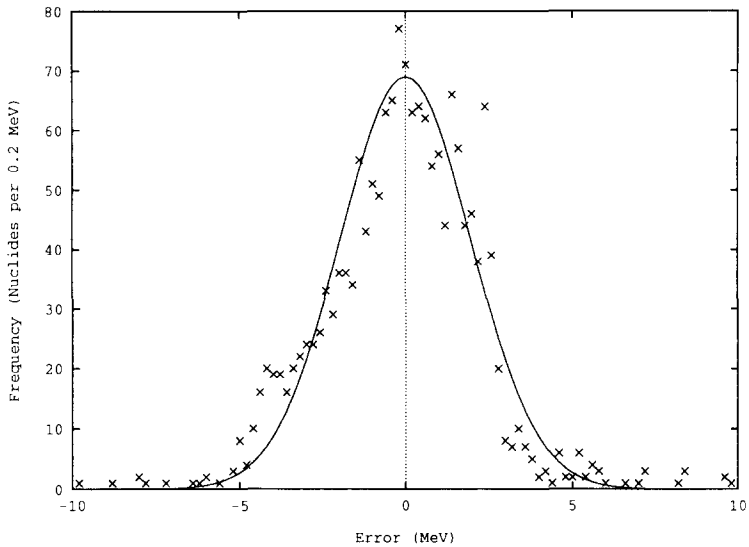

Fig. 2. Check of the validity of key assumption (I) that the model errors satisfy $\epsilon_m^{(r)} \in N(0, \sigma_m^2)$. This example refers to the training set in the case 1(10)4 of Table 3.

Table 5
Application of the "cleanprop" algorithm to realistic modeling of experimental mass data

Net	η	σ(train)	$\sigma^{[o]}$(train)	σ(test)	$\sigma^{[o]}$(test)
I	0.5	0.996	1.003	1.483	1.491
IIa	0.5	0.912	0.932	1.174	1.179
IIb	0.25	0.802	0.816	1.092	1.094
IIc	0.1	0.767	0.782	1.060	1.062

A hybrid (h) binary-analog input coding scheme is used, with each of Z and N coded in binary by eight on-off neurons, and $A = N + Z$ and $N - Z$ coded as floating-point numbers by two dedicated analog neurons [16]. The database consists of 1654 current experimental nuclear mass-excess values with associated standard deviations $\sigma_e^{(r)} = \sigma_r$. The training set is a randomly selected subset of 1323 nuclides, the remaining 331 examples being used to test prediction. Networks are all of type $^h(18+10+10+10+1)_a[421]$. Net IIc evolves from IIb, and IIb from IIa, under training with the indicated learning rates η. Connections were not pruned. Quoted results for the conventional rms error measure $\sigma^{[o]}$ of Eq. (12) and the derived model error measure σ of Eq. (10) correspond to the minimum σ found during the 2000 runs at each learning rate. Units of σ and $\sigma^{[o]}$ are MeV.

– was introduced in this paper solely as a vehicle for illustrating the performance of the cleanprop learning procedure. It may nonetheless be of interest to compare the best available neural-network models of atomic-mass systematics with state-of-the-art models afforded by conventional theory and phenomenology.

First, it should be understood that all approaches which show quantitative success in the reproduction of atomic masses over broad ranges in Z and N are to some degree phenomenological, in the sense that they involve some number of adjustable parameters. Fundamental theory (namely, quantum chromodynamics) has not yet reached the stage of tractability necessary for the calculation of the binding energy of even so simple a nucleus as the deuteron. "Effective" microscopic theories based on hadrons and hadronic interactions are also problematic, especially if they are pursued with relativistic dynamics. Even at the traditional level where nuclei are described as nonrelativistic collections of nucleons interacting via few-nucleon forces fitted to scattering data and few-body nuclei, significant quantitative progress has been limited to

light nuclides such as ^{6}Li and ^{16}O [48].

In the phenomenological approaches, certain parameters are adjusted to fit all of the measured masses or some subset of them. The physics embodied in this "training sample" (using the language of neural-network analysis) will be called the "internal physics" of the phenomenological model. In addition, models are generally guided or constrained by additional "external physics" (including, for example, relevant symmetry principles and what is known about semi-classical, pairing, and independent-particle aspects of nuclear energetics). The various models lie on a spectrum, positioned by the degree to which they are shaped by such "external" considerations. It is to be expected, and is generally the case, that those models enjoying a great deal of external physical input yield acceptable results with fewer adjustable parameters and are more successful in extrapolation outside the domain of the training sample to nuclei far from stability. The prime examples of this kind of model are the macroscopic-microscopic models of Möller, Nix, and collaborators [34,38,46], which appeal to the collective or macroscopic description provided by liquid-drop or droplet models, solve a one-body Schrödinger equation to incorporate single-particle degrees of freedom, and include pairing through a semi-microscopic calculation. Quite clearly, the neural-network mass models lie at the opposite end of the spectrum, in the sense that external physical input is minimal. This "purity" has a certain conceptual advantage in that neural-network studies can inform us on the extent to which the data, and only the data, serve to determine the underlying physical mass function $m(Z,N)$. On the other hand, it cannot be expected that neural-network models of the type considered here will be able to extrapolate far from the stable nuclei, even though they may show excellent predictive performance within the vicinity of the stable valley (i.e., for "homologous" nuclei, to borrow a term from protein chemistry [7]).

There are no purely statistical studies of the mass data (based on "standard" methods) with which the neural-network models can properly be compared. However, in the spectrum indexed by external physical content, the models of Jänecke and Masson [49] (JM) and Masson and Jänecke [50] (MJ) occupy positions intermediate between the neural-network models and the macroscopic-microscopic models of Möller and Nix, and their predictive power away from stability (their "extrapability") is also intermediate [45]. The Masson-Jänecke (JM and MJ) models are classified as "local," in that they employ local (in Z and N) relations among nuclidic masses to determine local values of the coefficients that specify the model [45]. (Models whose parameters are not subject to local adjustment – such as those constructed in the macroscopic-microscopic framework – are called "global.") The JM model (with 928 parameters) is based on the transverse Garvey-Kelson mass relations and includes relatively little external physics, whereas there is significantly more external physical content in the MJ model (with 471 parameters), which implements local relations in the more elaborate form of an inhomogeneous partial difference equation with higher-order isospin contributions.

For perceptive comparisons of a wide variety of mass models, particularly with a view to extrapability, the reader is referred to surveys by Haustein [44], Möller and Nix [38], and Borcea and Audi [45]. More to the point, the strengths and weaknesses of current neural-network analyses of atomic-mass data (principally in regard to interpolation versus extrapolation) have been examined in Ref. [16], where a quantitative comparison is made between neural-net results and results from the Möller-Nix macroscopic-microscopic approach. This study bears out the statements made above concerning the niche occupied by neural-network modeling.

The numerical comparison presented in Ref. [16] is updated and augmented in Table 6. As explained more fully below, the neural-net models (taken from Refs. [16,42]) are all of multilayer feedforward architecture and were created by supervised training algorithms of the backpropagation type. Traditional analyses of global character are represented by the highly refined macroscopic-microscopic finite-range droplet models of Möller et al. [46] (see also Möller and Nix [38]); and local treatments by the models of Jänecke and Masson [49] and Masson and Jänecke [50].

The databases used are the (mostly experimental) ground-state masses listed in the 1991 Brookhaven table (B), the 1986 experimental masses (cf. Ref. [51]), the 1992 experimental masses (cf. Ref. [52]), and a special database formed by Möller and Nix (MN) [38]. The last of these consists of the 1323 "old" (O) experimental masses to

Table 6
Comparison of neural-network models [16,42] of atomic-mass data with selected phenomenological models based on traditional nuclear theory

Net type or model		Learning mode		Prediction mode	
$^{c}(I+H_1+\cdots+H_{\mathcal{L}}+O)_{c'}[P]$	Data	Nuclides	$\sigma^{[o]}$	Nuclides	$\sigma^{[o]}$
$^{h}(18+10+10+10+1)_a[353]\star$	B	1719(R)	0.629	572(R)	0.736
$^{h}(18+10+10+10+1)_a[363]\star$	B	1719(R)	0.564	572(R)	0.725
$^{s}(4+57+1)_a[347]\diamond$	B	1719(R)	0.627	572(R)	0.771
Jänecke & Masson [928]	1986	1535	0.343	–	–
Masson & Jänecke [471]	1986	1504	0.346	–	–
Möller et al. [38]	1992	1654	0.681	–	–
$^{h}(18+10+10+10+1)_a[345]\star$	1992	1323(R)	0.525	331(R)	0.739
Möller et al. [38]	MN	1323(O)	0.673	351(N)	0.735
$^{h}(18+10+10+10+1)_a[421]$	MN	1323(O)	0.828	351(N)	5.981
$^{s}(4+40+1)_a[245]\diamond$	MN	1323(O)	1.068	351(N)	3.036

Quality of fit (Learning mode) and generalization capability (Prediction mode) are gauged by the rms error $\sigma^{[o]}$, in MeV. (The rms errors quoted for the two Masson-Jänecke models [49,50] are taken from the Haustein survey [44].) The number of input parameters for each model is indicated in square brackets as part of the model designation (for the models of Möller et al. [46] these are not all adjustable). See text for explanation of databases.

which the 1981 Möller-Nix model was fitted, together with 351 "new" (N) experimental mass values added since 1981. (It will be relevant that more than 90% of the 351 test nuclei of the MN database lie on or beyond the edges of the 1981 data set, as viewed in the (N,Z) plane, and many of these "outliers" are several grid points removed from any of the training examples.) The first three neural-network models correspond to the Brookhaven database, which was divided at random (R) into training and test sets of 1719 and 572 nuclides, respectively. The fourth neural-net model was trained on 1323 nuclides selected at random (R) from the 1992 data set used to determine the first of the two listed models of Möller et al., and tested on the remaining 331 examples (these are the same training and test sets as adopted in Application 2 above – cf. Table 5). The last two network models and the second model of Möller et al. were fitted to and tested on the MN training/test sets.

The networks marked with a diamond or a star symbol in Table 6 represent the culminations of two different lines of development of layered, feedforward neural-net models of the empirical mass data. The starred networks [16] employ the same redundant "hybrid" (h) input coding scheme as the nets of Table 5, and the same analog (a) output coding prescription. The rationale underlying the h scheme is that binary encoding of Z and N makes explicit their integral (quantal) character, which is essential to pairing and shell effects, while analog encoding of A and $N-Z$ facilitates the modeling of semiclassical effects described by the liquid-drop model (cf. Eq. (21)). The $\star$ networks derive from complex sequences of on-line backpropagation runs. Features going beyond vanilla backprop included (i) scheduled reductions of the learning rate, (ii) more frequent exposure to hard-to-learn patterns in the late stages of learning, and (iii) successive episodes in which connections judged unimportant (based on their effect on the mean-square cost function) were deleted – "pruned" – and the diluted network then retrained. This third refinement improves parametric efficiency and often improves generalization performance as well.

The nets labeled with a diamond were constructed by Kalman [42]. Attention was restricted to three-layer networks, but direct connections ("skip connections") were inserted between the input units and the single linear output unit that delivers the computed value for the mass excess. The transfer function $a_l = g(u_l) = \tanh(u_l) \in [-1,1]$ was used for hidden units [53], rather than the logistic function of Eq. (22). Determination of weights was carried

out within a Powell-update conjugate-gradient optimization routine in which the usual mean-square error measure was replaced by the better-behaved Kalman-Kwasny cost function [54]

$$E_{KK} = \frac{1}{2}\sum_r \frac{(t^{(r)} - a^{(r)})^2}{1 - (a^{(r)})^2} \,. \tag{25}$$

The raw input patterns consisted of the proton and neutron numbers Z and N, regarded as floating-point numbers, together with their "parities" $p(Z)$ and $p(N)$, with $p(x) = -1$ for x even and $+1$ for x odd. These patterns were preprocessed via singular-value decomposition [55] (thus the s designation for the input coding scheme); an affine transformation was then introduced to adjust the resulting components of the input-pattern matrix to lie in the interval $[-1, +1]$. The transformed patterns were fed into four input units. The $\star$ and $\diamond$ networks of Table 6 trained with Brookhaven data are directly comparable. It is interesting, and perhaps significant, that these finely tuned networks show very similar performance in spite of the different architectures and training procedures involved.

The last two network entries and the second Möller et al. listing also refer to common training and test sets (corresponding in particular to the nonrandom MN subdivision into "old" and "new" nuclei) and are therefore also directly comparable. The principal weakness of the neural-network models is quite evident in this comparison: extrapolation to outliers is poor. (It should be noted that the 421-parameter 5-layer net was obtained via vanilla backprop; hence some improvement in generalization might be achieved through cycles of pruning and retraining.) By contrast, when the test set is predominantly embedded within the (N, Z) domain encompassed by the training sample, and the task is predominantly one of interpolation, predictive performance can be excellent – as demonstrated on both the Brookhaven and 1992 data sets, for which $\diamond$ and/or $\star$ networks show small rms errors on the test samples, similar in size to the rms training errors.

The collective implication of the numerical studies summarized in Table 6 is that the current generation of neural-network models of the atomic-mass table can be competitive within the class of mass models with little overt theoretical input and (concomitantly) a relatively large number of adjustable parameters. However, lack of external physical constraints that both reduce the dimensionality of the fit and enhance extrapability puts the neural-net models at a disadvantage relative to "theoretically thick" models like those of Möller, Nix, and collaborators [38,46] in prediction of masses far from the valley of beta stability. Physically motivated regularization techniques, autoregression strategies, and advanced architectures offer some prospects for reducing this disadvantage.

Finally, we may quote supplementary results from Möller et al. [46] which relate to the cleanprop analysis of experimental mass data that was presented in Section 3.2. The values of the model error parameter σ corresponding to the three rms errors 0.681, 0.673, and 0.735 MeV given in Table 6 for the models of Ref. [46] are, respectively, 0.669, 0.671, and 0.686 MeV. Again it is seen that the discrepancy between σ and $\sigma^{[o]}$ is small in this problem, although it is certainly significant for the "new" nuclei.

4. Conclusions and prospects

4.1. Summary: development and application of cleanprop

A modified backpropagation procedure has been developed for training layered feedforward neural networks on a set of data points r for which the target or dependent variable is subject to noise characterized by uncertainty estimates σ_r. The procedure follows the obvious dictum that the noisier examples, with larger σ_r, should be accorded less influence in determination of the connection weights that define the network model. This dictum has been realized quantitatively through application of the maximum-likelihood criterion (MLC) in a form that allows for nonzero model error. The extended MLC leads naturally to replacement of the usual mean-square cost function of backpropagation by a new objective function and a procedure for estimating a new measure σ of the model error. The estimated model error provides a more realistic indication of the accuracy of the network's approximation to the true, uncontaminated mapping than does the conventional rms error $\sigma^{[o]}$ with respect to the noisy targets. The revised training scheme is easy to implement: it departs from standard on-line back-

propagation only in (i) the introduction of a pattern-specific learning rate $\eta/[1+(\sigma_r/\sigma)^2]$ and (ii) iterative re-determination of the model error measure σ after each epoch. The maximum-likelihood treatment assumes that the errors made by the model, relative to the true mapping, are normally distributed with zero mean and standard deviation σ, and that the errors in the targets corresponding to given instances r of the map (the experimental or simulation errors, in typical scientific applications) are normally distributed with zero mean and standard deviations σ_r. As necessary, these assumptions can be relaxed and the approach extended to problems with more complicated or less orthodox statistical properties.

In specific applications to the modeling of atomic masses, the new training algorithm (called "cleanprop") has proven to be very effective in suppressing the harmful effects of noise on the modeling process. Computational evidence has been presented which demonstrates that networks trained with cleanprop are capable of building satisfactory models of the true mapping even at high noise levels, provided the function to be approximated is reasonably smooth. It is anticipated that the cleanprop algorithm will be useful in a wide range of neural-network applications in which a statistical characterization of database errors is available.

4.2. Neural networks as computational aids in science

We close with some general remarks concerning the utility of artificial neural networks in scientific applications. The initial reception of this new computational approach in some sectors of the scientific community was perhaps too enthusiastic and not sufficiently critical. There was a tendency on the part of practitioners to oversell the powers of neural-network or "connectionist" solutions relative to conventional techniques – where conventional techniques can include both traditional theory-rich modeling and established statistical methods. The field is now in a correction phase. Second thoughts about the effectiveness of neural nets as tools for solving problems in physics and chemistry have been abundantly articulated in a recent paper in this journal by Duch and Diercksen [32]. These authors identify a number of misuses of neural nets and attempt to penetrate the mystique of neural-network technology, describing in some detail its application to linear regression. They stress that artificial neural networks are best suited to situations where a high error rate is tolerable and the problem to be solved is mathematically ill-posed.

A number of the warnings and criticisms expressed in the Duch-Diercksen article are well taken. It is certainly inappropriate to use neural networks to do arithmetic or symbolic manipulation tasks, which are much more efficiently performed by sequential computer programs. The potential strengths of neural networks, which are inherently parallel structures, lie rather in pattern recognition or more broadly in statistical inference (including classification and function approximation). Neural-network algorithms also offer novel approaches to the solution of constrained optimization problems.

In fact, the vast majority of applications, in science or otherwise, play to these potential strengths of connectionist systems and primarily involve regression or classification by layered feedforward networks ("perceptrons") subjected to supervised training algorithms, with backpropagation as the industry standard. In presenting the results of such applications, one should be careful not to overinterpret what a network has actually accomplished. Reiterating an example cited in Ref. [32], one should not be deluded that ordinary feedforward network models embody the Schrödinger equation just because they are able to learn – by example – the correlations between the parameters of some textbook Hamiltonian and its lowest energy eigenvalue. Neural networks should not be expected to do the formal or conceptual work of human scientists.

Moving on to the question of technical utility: it can be a wasteful exercise to invoke the heavy machinery of neural networks to fit a simple function of a small number of variables. This is most obvious in situations where physical grounds dictate or motivate a simple global form for the function, whose parameters can be easily determined by a least-squares minimization. (The neural-network treatment of the liquid-drop problem given in Section 3.1 is an obvious exception, since it was intended only as a well-controlled test of the cleanprop algorithm.) More generally, neural-network modeling may be an inefficient choice in low-dimensional problems when the familiar method of linear expansion in a set of "basis" func-

tions (including, as special cases, Fourier expansion, spline interpolation, and polynomial fits) would give an accurate representation [32].

Still, the pendulum should not swing too far in the direction of downgrading neural-network approaches. In problems where the input vector is of high dimension and/or traditional linear regression approaches prove inadequate, multilayer neural networks may have much to offer, *along with* other sophisticated methods of statistical analysis that have come on the scene in the last two or three decades. Neurally inspired approximation systems occupy a valid place in the taxonomy of nonparametric estimation [56,57] (see especially the authoritative survey by Friedman in Ref. [57]). Moreover, it is generally accepted that in some problem domains multilayer perceptrons can compete effectively with other, putatively more conventional, nonparametric nonlinear regression procedures such as projection pursuit and recursive partitioning techniques (notably CART and MARS) [56–58].

In passing judgment on neural networks, it should be understood that all approaches to nonlinear nonparametric regression must contend with the "curse of dimensionality." In high-dimensional spaces, one typically faces the problem that inordinately large training samples are needed to ensure good generalization – the dilemma of poor extrapability is hardly unique to backpropagation networks. (This commonality has the benefit that statistical methods already developed for dealing with this problem can be transplanted to neural-network modeling.)

It is well to reiterate that any universal method for function approximation (whether based on multilayer perceptrons or some more conventional treatment) has its own distinctive strengths and weaknesses; hence its performance relative to other universal methods may be very sensitive to the choice of problem. Another consideration that must be kept in mind when evaluating a critical numerical comparison of methods is that the author of the comparison will usually be implementing the different methods with different levels of skill [57]. Even if, in fair competition, a neural-network method proves not to be the optimal choice for speed and accuracy in a given application, it may still be preferred for practical reasons of convenience, such as readily available and easily adaptable software.

It is also important not to define artificial neural networks too narrowly when weighing their practical value. The power and flexibility of these systems can be substantially enhanced by such elaborations [3] on the usual perceptron theme as adaptive architectures (growing or pruning a net to fit the problem), recurrent connectivity, and hybrid supervised/unsupervised learning schemes. In particular, the latter schemes may invoke Gaussian transfer functions (for example, see Ref. [59]), a special example of radial basis functions [60].

Surely, one need not tie multilayer perceptron applications to the logistic sigmoid choice (22) for the neuronal transfer function (cf. [32]). (We hasten to point out that the cleanprop algorithm certainly transcends this choice.) Quite obviously, the transfer function should be adjusted to the specific needs of the problem at hand.

As a methodology for classification or function approximation in scientific problems, computational analysis based on neural networks is expected to prove most valuable in applications for which (i) the data set is large and complex, (ii) there is as yet no coherent theory of the underlying phenomenon, or quantitative theoretical explication is impractical, and (iii) no simple global model (involving simple parametrized functions) is available. Multilayer feed-forward nets, when taught by example, are capable of recognizing higher-order correlations in the input data and making appropriate decisions or computations based on their presence. In this sense, a successful network model could then be considered to embody fundamental features of the underlying phenomenon. Neural-network systems that arguably fall into this category include those developed for discrimination between star and galaxy images in optical sky survey plates [12], for the prediction of the branching probabilities for nuclear decay modes [17], and for the mapping of donor and acceptor sites in human genes [61]. On similar grounds, experimental high-energy physics has been considered a natural domain for application of neural-network hardware and software to data acquisition and analysis [9–11], not least because of the possibility of fast on-line implementation in environments with prodigious event rates.

Acknowledgements

This research has been supported in part by the U. S. National Science Foundation under Grant No. PHY-9307484 and by the BASF Aktiengesellschaft and the Studienstiftung des deutschen Volkes through a post-doctoral fellowship award to KAG. We are grateful to P. Möller and J. R. Nix for very helpful communications. It was their earlier work on the use of experimental mass uncertainties in determining model parameters that stimulated the present investigation. We extend thanks to B. L. Kalman for many valuable discussions and for permission to quote his mass-model results in advance of publication, and to E. Barnard for informative conversations. JWC has enjoyed the hospitality of the Theoretical Physics Institute of the University of Coimbra and the Physics Division of Argonne National Laboratory during important stages of this work.

References

[1] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vols. 1 and 2 (MIT Press, Cambridge, MA, 1986).

[2] B. Müller and J. Reinhardt, Neural Networks – an Introduction (Springer Verlag, Heidelberg, 1990).

[3] J. Hertz, A. Krogh and R.G. Palmer, Introduction to the Theory of Neural Computation (Addison-Wesley, Redwood City, CA, 1991).

[4] J.W. Clark, Phys. Med. Biol. 36 (1992) 1259.

[5] P.D. Wasserman, Advanced Methods in Neural Computing (Van Nostrand Reinhold, New York, 1993).

[6] S. Haykin, Neural Networks: A Comprehensive Foundation (McMillan, New York, 1994).

[7] N. Qian and T.J. Sejnowski, J. Mol. Biol. 202 (1988) 865.
H. Bohr, J. Bohr, S. Brunak, R.M.J. Cotterill, B. Lautrup, L. Nøskov, O.H. Olsen and S.B. Petersen, FEBS Letters 241 (1988) 223.
H. Bohr, J. Bohr, S. Brunak, R.M.J. Cotterill, H. Fredholm, B. Lautrup and S.B. Petersen, FEBS Letters 261 (1990) 43.
G.L. Wilcox, M. Poliac and M.N. Liebman, Tetrahedron Comput. Meth. 3 (1990) 191.
F.H. Stillinger, T. Head-Gordon and C.L. Hirschfeld, Phys. Rev. E 48 (1993) 1469.
T. Head-Gordon and F.H. Stillinger, Phys. Rev. E 48 (1993) 1502.

[8] J. Zupan and J. Gasteiger, Analytica Chimica Acta 248 (1991) 1; Neural Networks for Chemists: an Introduction (VCH, Weinheim, 1993).

[9] B. Denby, Comput. Phys. Commun. 49 (1988) 429.
C. Peterson, Nucl. Instr. Methods A279 (1989) 537.
L. Lönnblad, C. Peterson and T. Rögnvaldsson, Comput. Phys. Commun. 70 (1992) 167.
C. Bortolotto, A. de Angelis, N. de Groot and J. Seixas, Int. J. Mod. Phys. C 3 (1992) 733.
C. Peterson, T. Rögnvaldsson and L. Lönnblad, Comput. Phys. Commun. 81 (1994) 185.

[10] O. Benhar, C. Bosio, P. del Giudice and E. Tabet, Neural Networks: From Biology to High Energy Physics (ETS Editrice, Pisa, 1991).

[11] Proceedings of the Second Elba Workshop on Neural Networks: From Biology to High Energy Physics, Int J. Neural Systems, Suppl. Issue (1993).

[12] S.C. Odenwahn, E.B. Stockwell, R.L. Pennington, R.M. Humphreys, and W.A. Zumach, Astrophys. J. 103 (1992) 318.

[13] J.R.P. Angel, P. Wizinowich, M. Lloyd-Hart and D. Sandler, Nature 348 (1990) 221.
D.G. Sandler, T.K. Barrett, D.A. Palmer, R.Q. Fugate and W.J. Wild, Nature 351 (1991) 300.
M. Lloyd-Hart, P. Wizinowich, B. McLeod, D. Wittman, D. Colucci, R. Dekany, D. McCarthy, J.R.P. Angel and D. Sandler, Astrophys. J. 390 (1992) L41.

[14] K.L. Peterson, Phys. Rev. A 41 (1990) 2457; 44 (1991) 126.

[15] S. Gazula, J.W. Clark and H. Bohr, Nucl. Phys. A 540 (1992) 1.

[16] K.A. Gernoth, J.W. Clark, J.S. Prater and H. Bohr, Phys. Lett. B 300 (1993) 1.

[17] K.A. Gernoth and J.W. Clark, Neural Networks 8 (1995) 291.

[18] J.W. Clark, K.A. Gernoth and M.L. Ristig, in: Condensed Matter Theories, vol. 9, eds. J.W. Clark, A. Sadiq and K.A. Shoaib (Nova Science Publishers, Commack, NY, 1994) p. 519.

[19] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis (Wiley, New York, 1973).

[20] C.H. Anderson and E. Abrahams, in: Proceedings of the IEEE First International Conference on Neural Networks, San Diego, June 1987, vol. 3, eds. M. Caudill and C. Butler (IEEE, New York, 1987), p. 105.

[21] A. Lansner and Ö. Ekeberg, Int. J. Neural Systems 1 (1989) 77.

[22] H. White, Neural Comput. 1 (1989) 425.

[23] E. Barnard and D. Casasent, IEEE Trans. Syst., Man, Cybern. 19 (1989) 1030.
E. Barnard, IEEE Trans. Neural Networks 3 (1992) 1026.

[24] D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley and B.W. Suter, IEEE Trans. Neural Networks 1 (1990) 296.

[25] E.A. Wan, IEEE Trans. Neural Networks 1 (1990) 303.

[26] M.D. Richard and R.P. Lippmann, Neural Comput. 3 (1991) 461.

[27] P. Stolorz, A. Lapedes and Y. Xia, J. Molec. Biol. 225 (1991) 363.

[28] W.L. Buntine and A.S. Weigend, Complex Systems 5 (1991) 603.

[29] M. Opper and D. Haussler, Phys. Rev. Lett. 66 (1991) 2677.

[30] D.J.C. MacKay, Neural Comput. 4 (1992) 415, 448.

[31] T.L.H. Watkin, A. Rau and M. Biehl, Rev. Mod. Phys. 65 (1993) 499.

[32] W. Duch and G.H.F. Diercksen, Comput. Phys. Commun. 82 (1994) 91.

[33] H. Cramér, Mathematical Methods of Statistics (Princeton University Press, Princeton, NJ, 1946).

[34] P. Möller and J.R. Nix, At. Data Nucl. Data Tables 39 (1988) 213.
[35] J. Mathews and R.L. Walker, Mathematical Methods of Physics (W.A. Benjamin, New York, 1970), pp. 387–395.
[36] R. von Mises, Mathematical Theory of Probability and Statistics (Academic Press, New York, 1964).
[37] E. Barnard, IEEE Trans. Neural Networks 3 (1992) 232.
[38] P. Möller and J.R. Nix, in: Nuclei Far From Stability/Atomic Masses and Fundamental Constants 1992 (Bernkastel-Kues), IOP Conference Series No. 132, eds. R. Neugart and A. Wöhr (Institute of Physics Publishers, Bristol, 1993), p. 43; J. Phys. G 20 (1994) 1681.
[39] J.W. Clark, S. Gazula, K.A. Gernoth, J. Hasenbein, J. Prater and H. Bohr, in: Recent Progress in Many-Body Theories, vol. 3, eds. T.L. Ainsworth, C.E. Campbell, B.E. Clements and E. Krotscheck (Plenum, New York, 1992), p. 371.
[40] A. Bohr and B.R. Mottelson, Nuclear Structure, vol. 1 (W.A. Benjamin, New York, 1969).
[41] W.D. Myers and W.J. Swiatecki, Nucl. Phys. 81 (1966) 1.
[42] B.L. Kalman, private communication.
[43] S.M. Weiss and C.A. Kulikowski, Computer Systems that Learn (Morgan Kaufmann, San Mateo, CA, 1990).
[44] P.E. Haustein, At. Data Nucl. Data Tables 39 (1988) 185.
[45] C. Borcea and G. Audi, Report CSNSM-92.38, Orsay (1992); Rom. J. Phys. 38 (1993).
[46] P. Möller, J.R. Nix, W.D. Myers and W.J. Swiatecki, At. Data Nucl. Data Tables, in press.
[47] K. Hornik, M. Stinchcombe and H. White, Neural Networks 2 (1989) 359.
[48] J. Carlson, V.R. Pandharipande, B. Pudliner and R.B. Wiringa, to be published.
S.C. Pieper, R.B. Wiringa and V.R. Pandharipande, Phys. Rev. C 46 (1992) 1741.
S.C. Pieper and V.R. Pandharipande, Phys. Rev. Lett. 70 (1992) 2541.
[49] J. Jänecke and P.J. Masson, At. Data Nucl. Data Tables 39 (1988) 265.
[50] P.J. Masson and J. Jänecke, At. Data Nucl. Data Tables 39 (1988) 273.
[51] A.H. Wapstra, G. Audi and R. Hoekstra, At. Data Nucl. Data Tables 39 (1988) 281.
[52] G. Audi and A.H. Wapstra, Nucl. Phys. A 565 (1993) 1.
[53] B.L. Kalman and S.C. Kwasny, in: Proceedings of the International Joint Conference on Neural Networks, vol. IV, Baltimore, MD (1992), p. 578.
[54] B.L. Kalman and S.C. Kwasny, in: Proceedings of the International Joint Conference on Neural Networks, vol. 2, Seattle, WA (1991), p. 49.
[55] B.L. Kalman, S.C. Kwasny and A. Abella, in: Proceedings of the World Congress on Neural Networks, vol. IV, Portland, OR (1992), p. 578.
[56] M. Smith, Neural Networks for Statistical Modeling (Van Nostrand Reinhold, New York, 1993).
[57] V. Cherkassky, J.H. Friedman and W. Wechsler, eds., From Statistics to Neural Networks. Theory and Pattern Recognition Applications (Springer Verlag, Berlin, 1994).
[58] E. Barnard, in: Proceedings of Neural Networks for Physicists II (Theoretical Physics Institute, Minneapolis, 1992); and private communication.
[59] J. Moody and C. Darken, Neural Comput. 1 (1989) 281.
[60] M.J.D. Powell, in: Algorithms for Approximation, eds. J.C. Mason and M.G. Cox (Clarendon Press, Oxford, 1987).
[61] S. Brunak, J. Engelbrecht and S. Knudsen, J. Mol. Biol. 220 (1991) 49.